REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 07/17/00	3. REPORT TYPE AND DATES COVERED Final Progress Report 05/01/96 - 04/30/00

4. TITLE AND SUBTITLE
Adaptive Algorithms for Active Noise and Vibration Controll

5. FUNDING NUMBERS
DAAH04-96-1-0085

6. AUTHOR(S)
Marc Bodson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical Engineering
University of Utah
50 S Central Campus Dr Rm 3280
Salt Lake City, UT 84112-9206

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 34533.11-MA

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The objective of this project was to develop and analyze advanced adaptive schemes for the active attenuation of sound and vibration. Implementations of algorithms such as the filtered-X LMS algorithm were proposed that require fewer computations than the standard implementation. Delay compensation techniques were also developed to combat performance-reducing effects due to delay within the parameter updates. In particular, the extension of the method to adaptive echo cancellation was investigated. Progress was also made on adaptive algorithms for the rejection of periodic disturbances without measurement of the disturbance or of its frequency. A direct approach and an indirect approach were studied, and the resulting algorithms were tested successfully on an active noise control testbed. Disturbances with multiple harmonics and multi-channel systems were both considered.

14. SUBJECT TERMS
Active noise control, active vibration control, adaptive algorithms, delay compensation, LMS algorithm, multi-channel systems, narrowband control, periodic disturbances, sinusoidal disturbances, phase-locked loops, feedback control.

15. NUMBER OF PAGES
182

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

20001122 132

Final Progress Report – Army Research Office
Adaptive Algorithms for Active Noise and Vibration Control

Marc Bodson*
Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112
(801) 581 8590 – bodson@ee.utah.edu
Fax: (801) 581 5281

July 17, 2000

Funding Number: DAAH-04-96-1-0085
ARO Proposal Number: 34533-MA
Period Covered: 05/01/96-04/30/00

1. Scientific Progress and Accomplishments

1.1 Objectives

The objective of the project was to advance the state-of-the-art in adaptive methods for disturbance cancellation. Two closely-related applications were considered: active noise control and active vibration control. Improvements in adaptive methods for these applications are of potential benefit to several systems used by the Army, and helicopters in particular.

A single-channel active noise control system is shown on Fig. 1. A microphone senses the noise at a location to be made quiet, and a digital signal processor (DSP) uses the signal to produce a sound wave that will reduce the noise. The DSP implements a control algorithm that is typically adaptive, and must be computed at a high sampling rate. The focus of the project was this algorithm, or software, that is coded on the DSP. Performance and computational requirements were driving forces of the investigation.

The first track of the project was aimed at studying design and implementation issues for adaptive methods as applied to multi-channel systems. As opposed to Fig. 1, practical active noise

*Dr. Scott Douglas contributed some portions of this report and is now with the Department of Electrical Engineering, Southern Methodist University, Dallas, TX 75275. Tel.: (214) 768 4595. E-mail: douglas@seas.smu.edu.

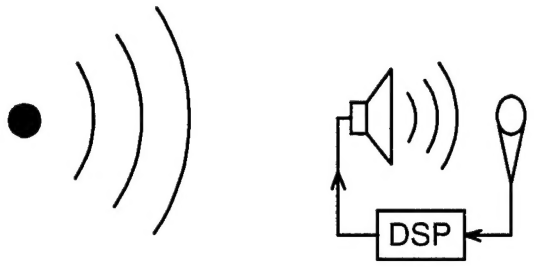

Figure 1: Single-channel active noise control system

control systems are typically *multi-channel* systems, that is, rely on several speakers and microphones to perform the noise cancellation. Indeed, the attenuation of noise over a significant area of space usually requires the use of multiple sound sources. For vibration control, multiple actuators are necessary when several degrees of freedom are involved. In the project, novel implementations of commonly-used adaptive control structures were investigated that required significantly fewer computations than existing structures. Delay compensation techniques were also developed that enhanced the performance of certain algorithms in the presence of large delays, without incurring the usual penalties in terms of computations and memory accesses.

The second track of the project investigated new algorithms tailored to the rejection of periodic disturbances. Such disturbances are common because of numerous applications where rotating machines are the source of unwanted vibration and noise. Two approaches were studied: an indirect approach which combined frequency estimation with adaptive disturbance rejection, and a direct approach which integrated frequency estimation and disturbance cancellation. Both approaches were developed for situations where the frequency of the disturbance varied significantly with time and they eliminated the often-found requirement of a sensor placed on or close to the source of the disturbance. The algorithms were evaluated in experiments on an active noise control testbed developed under the grant at the University of Utah.

1.2 Accomplishments in Multi-channel Systems

The filtered-X LMS algorithm, commonly used in feedforward multi-channel active noise control systems, is difficult to implement for systems with more than a few input sensors, output actuators, and error sensors, due to the large number of computations required. Since large-area quieting demands a large number of controller channels, this difficulty limits the usefulness of the algorithm. A careful study of the computations required by the standard implementation of the algorithm indicated that the bulk of the complexity lied in the controller coefficient updates. With this observation in mind, a computationally-efficient, fast convolution algorithm was developed. The method computes quantities that are needed by the filtered-X LMS controller and calculates correction terms at each time instant to produce the exact controller output signals. The complexity of the

new scheme was much reduced over that of the standard implementation: reductions of 90% or more in the overall computational and memory usage were found to be common [1], [8].

The results obtained with the filtered-X LMS algorithm encouraged the application of the technique to other algorithms. Computationally-efficient, block convolution methods were developed for several algorithms popular in single-channel feedforward active noise control. The methods employed a two-step calculation process and yielded behavior that was identical to the original algorithm implementation, but with a reduced computational complexity. In general, reductions in the number of multiplications of the order of 25% were found to be common [4], [7]

In the area of stability and parameter convergence, an efficient delay compensation technique was developed for the filtered X-LMS algorithm that effectively removed the coefficient delays in the error signals that were caused by the plant [3]. The resulting algorithm behaved like the LMS algorithm, and since the behavior of the LMS algorithm was well-understood, the behavior of this delay-compensated system could be easily predicted from existing results in adaptive filter theory. This feature eased the selection of adaptation parameters for the stability of the system. The delay compensation technique was expanded into a family of delay compensation methods for gradient-based algorithms with feedback delay . These techniques could be used to mitigate the algorithm instabilities caused by plants that exhibit a large amount of delay and whose characteristics vary with time. In particular, the numerically-robust techniques were extended to useful signal processing algorithms such as the sliding discrete Fourier and cosine transforms [2], [5]. Further a significant breakthrough was obtained when the methods were extended to the case of time-varying plants [6].

The methods can be combined in several ways to obtain algorithms that do not compute the plant-filtered input signal, thus avoiding the memory allocation and accesses that can make algorithms difficult to implement on typical digital signal processor platforms. Reducing the number of memory accesses is important for some adaptive signal processing problems, notably acoustic echo cancellation. Using the delay-compensation method, four novel implementations of the normalized least-mean square (NLMS) algorithm with decorrelation filters were developed for acoustic echo cancellation [9]. For short decorrelation filter lengths, the complexity of the new implementations is nearly the same as the complexity of the uncompensated NLMS adaptive filter, as opposed to the conventional implementation of the compensation, which increases the complexity by about 50%. Two of the novel implementations also saved on memory use over the conventional implementation.

1.3 Accomplishments in Periodic Disturbance Rejection

Two approaches were proposed for the rejection of periodic disturbances with unknown frequency [11]. The first was an indirect algorithm where the frequency of the disturbance was estimated, and the estimate was used in an adaptive algorithm that adjusted the magnitude and phase of the input needed to cancel the effect of the disturbance. A direct algorithm was also investigated in which the frequency estimation and the disturbance cancellation were performed simultaneously.

Analyses were presented for both schemes and the results were found useful for the selection of the design parameters. Simulations were given that demonstrated the validity of the analytical results and the ability of the algorithms to reject sinusoidal disturbances of unknown frequency.

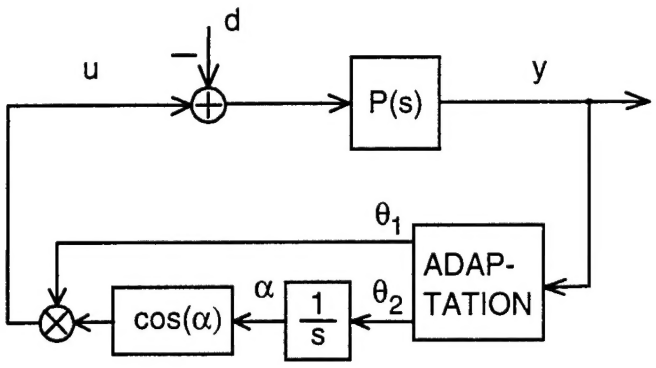

Figure 2: Direct approach for sinusoidal disturbance cancellation

The concept of the direct approach is shown in Fig. 2. The control system is similar, but distinct, from the well-known and highly successful phase-locked loops used in communication systems. A key tool in the analysis of the algorithm was a nonlinear time-invariant approximation of the control system that could be used for analysis and design. Simulations showed that the nonlinear approximation reflected very well the transient behavior of the scheme. Further, this approximation was useful to predict the ability of the algorithm to reject slowly-varying disturbances, which was validated in simulations [10]. In [16], the algorithm was analyzed in the presence of measurement noise, and estimates were obtained for the variance of the output and of the adaptive parameters. In both [10] and [16], simulations verified the validity of the analytical results, and demonstrated the excellent properties of the algorithm under non-ideal conditions.

A real-time implementation of the direct scheme in an active noise control task was successfully performed on a DSP system in the principal investigators' laboratory. The active noise control testbed had been developed under the grant with the help of undergraduate students associated with the project. The scheme provided a 25dB reduction of a sinusoidal tone of unknown frequency. The direct method was also extended to handle the case of multiple harmonics. The resulting algorithm is shown in Fig. 3 for the case of a fundamental and a third harmonic. An interesting feature of the algorithm was that the frequencies of the different components of the control signal were tied together, based on the knowledge of the relationship between the frequencies of the components of the disturbance. On the figure, this relationship is represented by the multiplying factor of "3". Laboratory experiments demonstrated the success of the algorithm in a practical environment, and the results appeared in a conference paper that has now been accepted for publication in a journal [14]. The scheme implemented an automatic tuning procedure that adjusted the controller parameters based on the identified frequency response of the plant, and the controller gains were

scheduled as functions of the estimated frequency of the disturbance.

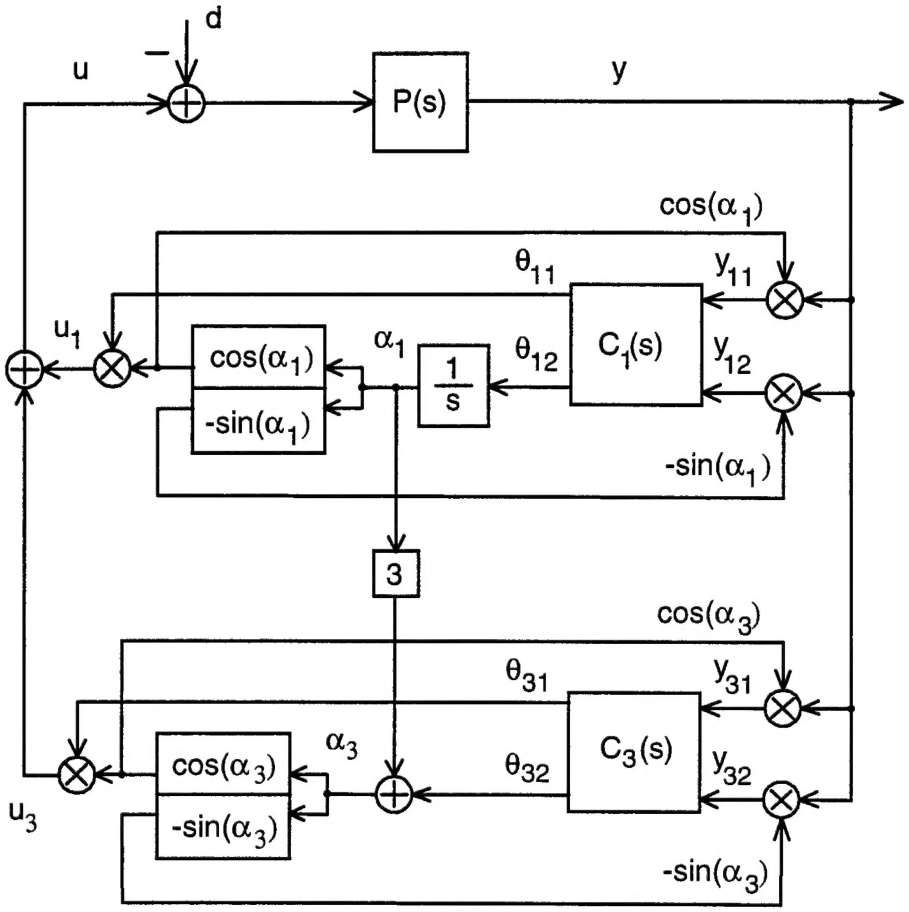

Figure 3: Direct algorithm for periodic disturbance cancellation

In [12], the direct algorithm was compared to an adaptive algorithm based on the internal model principle. This study was pursued in response to comments made by a reviewer of the original proposal for the grant. It was found that the algorithm based on the internal model principle had interesting advantages by guaranteeing global stability under ideal conditions, tracking of reference inputs, and stabilization of unstable plants. However, the algorithm was also found to suffer from convergence and robustness problems, and the assumptions that it required were found to make it poorly suited to high-order systems and systems with unknown time delays. It was concluded that, for active noise control applications, the advantages of the internal model principle algorithm were of little value, while its disadvantages made a practical implementation difficult if not impossible. For this reason, further work was not pursued along this direction.

In [15], a scheme proposed in a patent by Chaplin & Smith was analyzed. The paper demonstrated that the techniques of analysis developed under the grant could be applied to Chaplin &

Smith's algorithm, resulting in an ability to predict the stability properties of the algorithm, select the design parameters, and improve the overall dynamic response. It was also found that the direct approach under study in this project had distinguishing features and advantages over Chaplin & Smith's approach.

In the process of surveying various approaches to periodic disturbance rejection, a modification of the indirect approach was discovered that resulted in significant improvements. Earlier simulations of the indirect algorithm had shown convergence problems which made the scheme unattractive. However, simulations for the modified algorithm changed that picture [13]. The idea of the modified algorithm is shown in Fig. 4, and consists in subtracting from the plant output an estimate of the portion of the plant output that is due to the input signal.

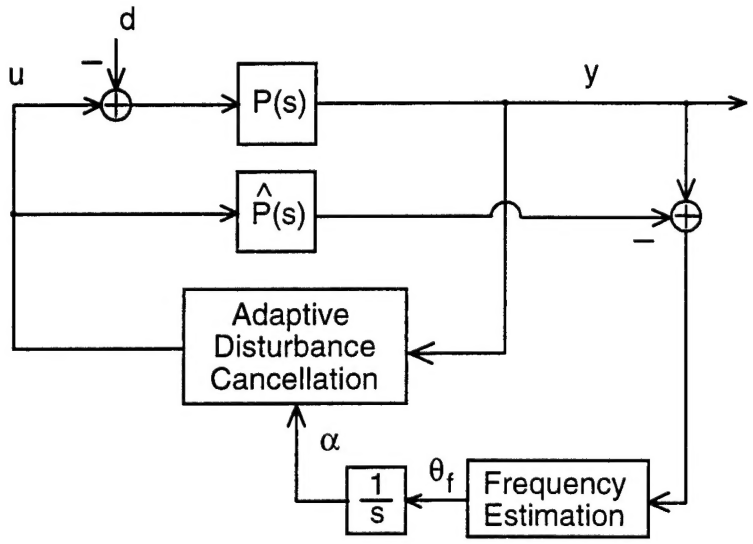

Figure 4: Indirect approach to periodic disturbance cancellation

Experiments in the lab using the (modified) indirect approach were very successful, with noise reductions by an order of magnitude obtained for a sinusoidal disturbance. The algorithm was extended to a multi-channel system and a conference paper discussing the results appeared in 1999 [17]. The microphone signals reported in the paper are shown on Fig. 5, and the frequency estimate is shown in Fig. 6.

Research along the direction of periodic disturbance cancellation has continued after the completion of the grant. Interestingly, a new frequency estimation algorithm was developed based on the direct scheme, and a conference paper is being prepared for the *2001 American Control Conference*. Our plan is to integrate this frequency estimation algorithm with the indirect adaptive scheme in order to obtain a multi-channel, multi-harmonic adaptive cancellation scheme.

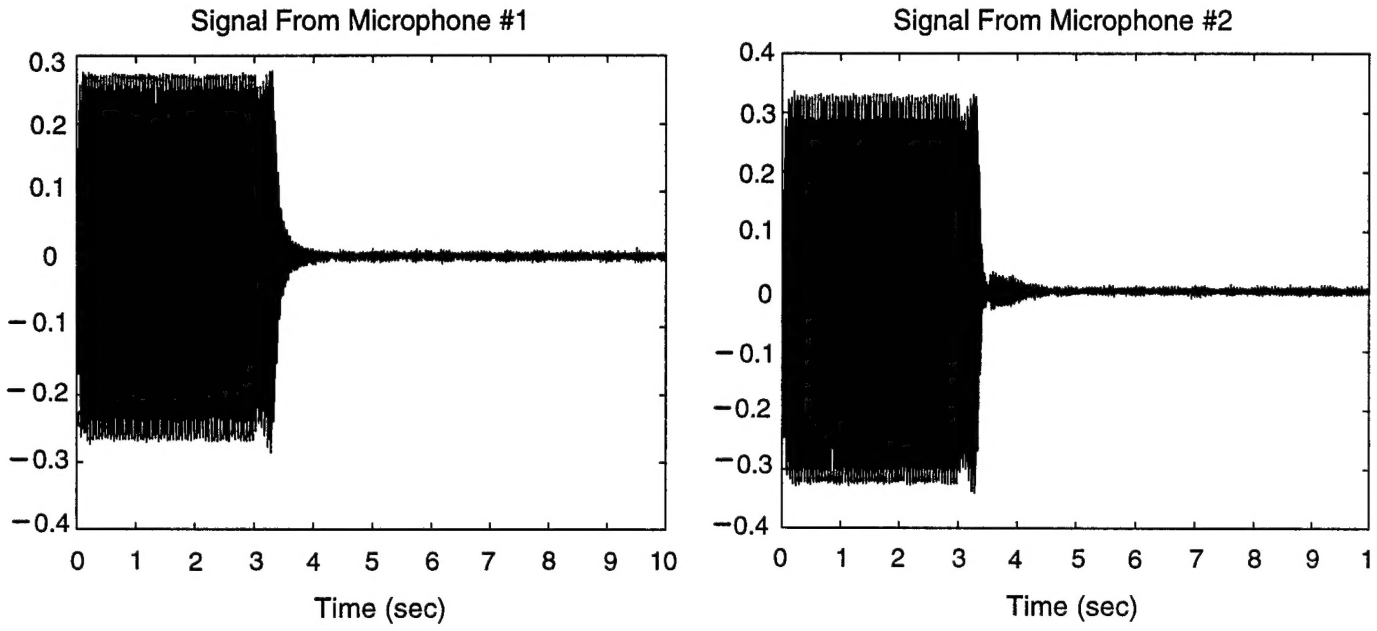

Figure 5: Microphone signals for multi-channel disturbance cancellation experiment

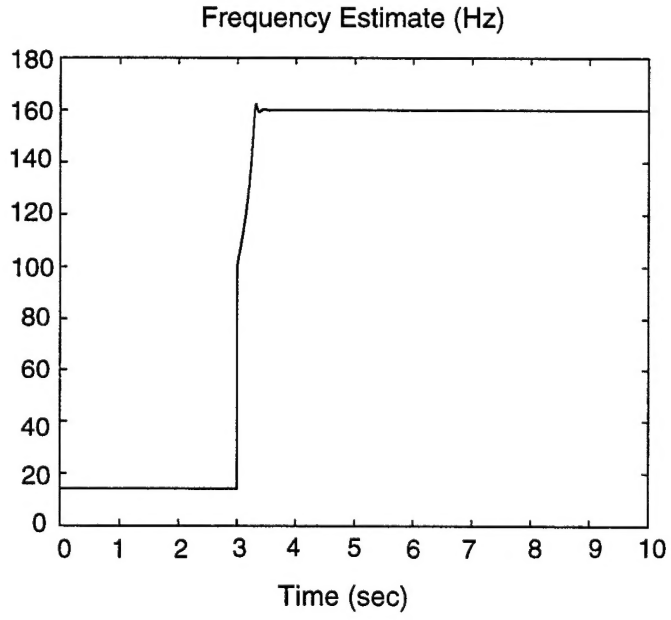

Figure 6: Frequency estimate for multi-channel disturbance cancellation experiment

2. List Of Manuscripts

[1] S.C. Douglas, "Reducing the Computational and Memory Requirements of the Multichannel Filtered-X LMS Adaptive Controller," *Proc. National Conf. Noise Control Engineering,* Philadelphia, PA, vol. 2, pp. 209-220, 1997.

[2] S.C. Douglas and J.K. Soh, "Numerically-Stable Implementations of the Sliding DFT, Sliding DCT, and Sliding-Window RLS Algorithms," submitted to *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing,* 1997.

[3] S.C. Douglas, "An Efficient Implementation of the Modified Filtered-X LMS Algorithm," *IEEE Signal Processing Letters,* vol. 4, no. 10, pp. 286-288, 1997.

[4] D.S. Nelson, S.C. Douglas, and M. Bodson, "Fast Block Adaptive Algorithms for Feedforward Active Noise Control," *Proc. National Conf. Noise Control Engineering,* Philadelphia, PA, vol. 2, pp. 197-208, 1997.

[5] S.C. Douglas and J.K. Soh, "A Numerically-Stable Sliding-Window Estimator and Its Application to Adaptive Filtering," *Proc. 31st Annual Asilomar Conf. on Signals, Systems, and Computers,* Pacific Grove, CA, vol. 1, pp. 111-115, 1997.

[6] S.C. Douglas and J.K. Soh, "Delay Compensation Methods for Stochastic Gradient Adaptive Filters," *Proc. 8th IEEE Digital Signal Processing Workshop,* Bryce Canyon, UT, paper no. 108, 1998.

[7] D.S. Nelson, S.C. Douglas, and M. Bodson, "Fast Exact Adaptive Algorithms for Feedforward Active Noise Control," Submitted to the *Int. J. Adaptive Control Signal Processing,* special issue on Acoustic Echo and Noise Control, R.W. Stewart and S. Weiss, editors, 1999.

[8] S.C. Douglas, "Fast Implementations of the Filtered-X LMS and LMS Algorithms for Multichannel Active Noise Control," *IEEE Trans. Speech and Audio Processing,* vol. 7, no. 4, pp. 454-465, 1999.

[9] J.K. Soh & S.C. Douglas, "Efficient Implementations of the NLMS Algorithm with Decorrelation Filters for Acoustic Echo Cancellation," *Proc. of the International Workshop on Acoustic Echo and Noise Control,* Pocono Manor, PA, pp. 164-167, 1999.

[10] M. Bodson and S.C. Douglas, "Rejection of Disturbances with a Large Sinusoidal Component of Unknown Frequency," *Proc. of the SPIE Symposium on Smart Structures and Materials,* San Diego, CA, vol. 2715, pp. 64-75, 1996.

[11] M. Bodson & S. Douglas, "Adaptive Algorithms for the Rejection of Sinusoidal Disturbances with Unknown Frequency," *Proc. of the 13th World Congress of the International Federation*

of Automatic Control, San Francisco, CA, vol. K, pp. 229-234, 1996. An expanded version appeared in *Automatica,* vol. 33, no. 12, pp. 2213-2221, 1997.

[12] M. Bodson and S.C. Douglas, "Narrowband Disturbance Rejection using Adaptive Feedback Algorithms," *Proc. of the SPIE Symposium on Smart Structures and Materials,* San Diego, CA, vol. 3039, pp. 45-56, 1997.

[13] M. Bodson, "Algorithms for Feedback Active Noise Control and Narrowband Disturbances," *Proc. of the ISMA23 (International Conference on Noise and Vibration Engineering),* Leuven, Belgium, vol. 3, pp. 1279-1286, 1998.

[14] M. Bodson, J. Jensen, & S. Douglas, "Active Noise Control for Periodic Disturbances," *Proc. of the American Control Conference,* Philadelphia, PA, pp. 2616-2620, 1998. Revised version accepted for publication in *IEEE Trans. on Control Systems Technology.*

[15] M. Bodson, "A Discussion of Chaplin & Smith's Patent for the Cancellation of Repetitive Vibrations," *Proc. of the IEEE Conference on Decision and Control,* Tampa, FL, pp. 1560-1565, 1998. Also appears in *IEEE Trans. on Automatic Control,* vol. 44, no.11, pp. 2221-2225, 1999.

[16] M. Bodson, "Performance of an Adaptive Algorithm for Sinusoidal Disturbance Rejection in High Noise," submitted for publication to *Automatica.* The paper is an expanded version of "Cancellation of Sinusoidal Disturbances with Unknown Frequency," that appeared in the *Proc. of the Ninth Yale Workshop on Adaptive and Learning Systems,* New Haven, CT, pp. 174-179, 1996

[17] B. Wu & M. Bodson, "Multi-Channel Active Noise Control for Periodic Disturbances," *Proc. of the IEEE Conference on Decision and Control,* Phoenix, AZ, pp. 4971-4975, 1999.

3. Scientific Personnel

Faculty: Marc Bodson and Scott C. Douglas.

Students supported: David Nelson, Jihee Soh, Biqing Wu, and Haiping Wu. David S. Nelson completed his M.S. thesis titled "Fast Exact Block Adaptive Algorithms for Feedforward Active Noise Control" in 1998 and joined L3 Communications Corp., Salt Lake City, UT. Jihee K. Soh completed her M.S. thesis at the University of Utah and joined the grant to work on her Ph.D. on the topic of delay compensation methods. She passed the Ph.D. qualifying examinations. Biqing Wu passed her Ph.D. qualifying examinations in Spring 1999 and has been working on her Ph.D. thesis on the topic of frequency estimation and active noise control. She is expected to defend her Ph.D. thesis proposal in August 2000. Haiping Wu worked on the design of adaptive systems for periodic disturbance cancellation using linear time-invariant equivalence. He defended his M.S. thesis proposal shortly after the end of the grant, in May 2000, and is expected to defend his

M.S. thesis in August 2000. Jonathan Jensen was supported briefly on the grant, but decided not to continue working on the project. His help on developing the testbed, however, was invaluable. Shayne Messerly and Cameron Ford completed senior projects on topics related to the grant without salary support from the grant, but using the active noise control testbed and supervision from the PI's.

4. Report of Inventions

A provisional patent entitled, "Method and apparatus for multi-channel active noise and vibration control," was filed. The costs of this application were paid by an outside commercial firm that may license the technology.

5. Technology Transfer

On March 24, 1997, Profs. Marc Bodson and Scott C. Douglas visited the Structural Acoustics Branch of Langley Research Center in Hampton, Virginia. In the morning, Drs. Bodson and Douglas held meetings with Dr. Richard Silcox and Dr. Karen Lyle of NASA Langley Research Center, in which they discussed their current work on active noise and vibration control algorithms. In the afternoon, they gave a joint talk entitled, "Novel Feedforward and Feedback Algorithms for Active Noise Control." Afterwards, Drs. Silcox and Lyle provided Drs. Bodson and Douglas a tour of the experimental facilities of the Structural Acoustics Branch at NASA Langley.

Approximately 20 researchers attended the talk, and the results stimulated a large number of questions and comments. Because a multichannel active noise control system was currently under development at NASA Langley, the researchers expressed a particular interest in the efficient implementation of the filtered-X LMS algorithm. Following the visit, Dr. Douglas was contacted by Randolph Cabell, a researcher at NASA Langley on leave from Virginia Polytechnic Institute and State University (Virginia Tech), who inquired about the efficient multi-channel filtered-X LMS algorithm developed under the Army Research Office grant.

On March 25, 1997, Drs. Bodson and Douglas visited the Thomas Lord Research Center of Lord Corporation, Cary, North Carolina. Lord Corporation is a manufacturer of adhesives, mounts, and active noise and vibration control systems. The visit to Lord Corporation was hosted by Drs. Steve Southward and Lane Miller. The talk generated significant interest in both the feedforward and feedback algorithms under development at the University of Utah. Afterwards, Dr. Steve Southward and Dr. Lane Miller of Lord Corporation gave a tour of the laboratories where active noise and vibration control research and development is underway. From this visit, it appeared that Lord Corporation had many advanced developments underway, and there was potential for future cooperation in the commercialization of active noise and vibration control systems for aeronautical applications.

NOISE-CON 97

The Pennsylvania State University
University Park, Pennsylvania
1997 June 15-17

REDUCING THE COMPUTATIONAL AND MEMORY REQUIREMENTS OF THE MULTICHANNEL FILTERED–X LMS ADAPTIVE CONTROLLER

Scott C. Douglas

Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112 USA

I. INTRODUCTION

Interest in active methods for the suppression of noise and vibration has grown recently, as evidenced by the numerous review articles and books that have appeared on the subject [1]–[9]. Although the potential for active noise and vibration control has long been recognized [10], successful implementations of these techniques have begun to appear only recently. Such success can be attributed to the rapid maturation of technology in three areas: (i) novel electroacoustic transducers, (ii) advanced adaptive control algorithms, and (iii) inexpensive and reliable digital signal processing (DSP) hardware. As advances in these areas are developed, active suppression of noise and vibration can be expected to find wider use in a number of commercial, industrial, and military applications. In our discussion, we focus on the algorithms used in multichannel active noise and vibration control systems as implemented in DSP hardware.

Perhaps the most popular adaptive control algorithm used in DSP implementations of active noise and vibration control systems is the filtered–X least-mean-square (LMS) algorithm. There are several reasons for this algorithm's popularity. Firstly, it is well-suited to both broadband and narrowband control tasks, with a structure that can be adjusted according to the problem at hand. Secondly, it is easily described and understood, especially given the vast background literature on adaptive filters upon which the algorithm is based [11, 12]. Thirdly, its structure and operation are ideally suited to the architectures of standard DSP chips, due to the algorithm's extensive use of the multiply/accumulate (MAC) operation. Fourthly, it behaves robustly in the presence of physical modelling errors and numerical precision effects caused by finite-precision calculations. Finally, it is relatively simple to set up and tune in a real-world environment.

Despite its popularity, the filtered–X LMS algorithm suffers from one drawback that makes it difficult to implement when a multichannel controller is desired: the complexity of the coefficient updates for the finite-impulse-response (FIR) filters within the controller in these situations is often much greater than the complexity of the input-output calculations. It is not unusual for the coefficient updates to require more than *ten times* the number of MACs needed to compute the outputs of the controller for fixed coefficient values, and the situation worsens as the number of error sensors is increased. For this reason, recent efforts have focused on ways to reduce the complexity of the filtered–X LMS algorithm in a multichannel context. Suggested changes include: (i) block processing of the coefficient updates using fast convolution techniques [13], (ii) partial

updating of the controller coefficients [14], and (iii) filtered-error methods [15, 16]. While useful, these methods often reduce the overall convergence performance of the controller, either because they introduce additional delays into the coefficient update loop or because they throw away useful information about the state of the control system. Such a performance loss may not be tolerable in some applications.

In addition to these computational difficulties, the multichannel filtered–X LMS algorithm also suffers from excessive data storage requirements. This algorithm employs filtered input signal values that are created by filtering every input signal by every output-actuator-to-error-sensor channel of the acoustic plant. The number of these terms can be an order-of-magnitude greater than the number of controller coefficients and input signal values used in the input-output calculations. As typical DSP chips have limited on-chip memory, system designers may be forced to use costly off-chip memory within their controller architectures that can further slow the operation of the system due to limits in input/output data throughput. While some of the aforementioned techniques for complexity reduction also have reduced memory requirements, the performance of the overall system is effectively limited by these methods.

In this paper, we present an alternative method for reducing the computational and memory requirements of the multichannel filtered–X LMS adaptive controller. Our solution is an alternative implementation of this system that is mathematically-equivalent to the original implementation, and thus it preserves the characteristic robust and accurate behavior of the filtered–X LMS algorithm. However, the complexity and memory requirements of the new implementation are significantly reduced over those of the original implementation, especially for controllers which a large number of channels. Moreover, since the filtered-input signals are never formed in our implementation, the excessive memory requirements of the original implementation are avoided.

In what follows, we first describe the alternate implementation of the filtered–X LMS algorithm in the single-channel case for simplicity, even though the implementation's computational savings are only realized in the multichannel case. As for mathematical notation, scalar variables are employed throughout to enable the algorithm's direct translation to processor code, and indices of parameter sets are for the most part lower-case versions of the variable designating the number of parameters; e.g. h_m for $1 \leq m \leq M$.

II. THE FILTERED-X LMS ALGORITHM

To simplify our discussion, we initially present the single-channel filtered–X LMS adaptive feedforward controller; the multichannel filtered–X LMS algorithm is described in Section IV. Figure 1 shows a block diagram of this system. In this system, a sensor placed near a noise source collects samples of the input signal $x(n)$ for processing by the system. This system computes an actuator output signal $y(n)$ using a time-varying FIR filter of the form

$$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l), \tag{1}$$

where $w_l(n)$, $0 \leq l \leq L-1$ are the controller coefficients and L is the controller filter length. The acoustic output signal produced by the controller combines with the noise signal as it propagates to the desired quiet region, where an error sensor collects the combined signal. We can model this error signal as

$$\epsilon(n) = d(n) + \sum_{m=-\infty}^{\infty} h_m y(n-m), \tag{2}$$

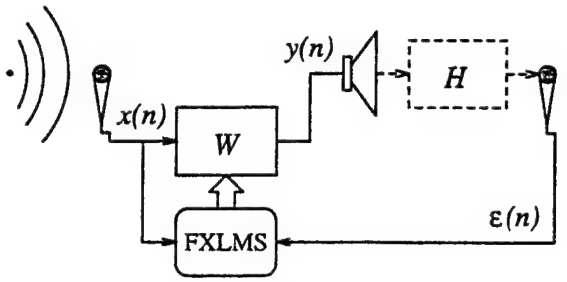

Figure 1: A feedforward filtered–X LMS adaptive controller.

where $d(n)$ is the unattenuated noise signal and h_m, $-\infty < m < \infty$ is the plant impulse response. Note that (2) is never computed as $\epsilon(n)$ is a measurement of a physical quantity.

The filtered–X LMS coefficient updates are given by

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)f(n-l), \tag{3}$$

where the filtered input sequence $f(n)$ is computed as

$$f(n) = \sum_{m=1}^{M} h_m x(n-m), \tag{4}$$

and M is the FIR filter length of an appropriate estimate of the plant impulse response. In practice, the values of h_m used in (4) are estimates of the actual h_m in (2) and are usually obtained in a separate estimation procedure that is performed prior to the application of control. For notational simplicity, we will not distinguish the differences in these two parameter sets in what follows.

A study of (1), (3), and (4) shows that the filtered–X LMS algorithm requires $2L + M + 1$ multiply/add operations and $2L + \max\{L, M+1\} + 1$ memory locations to store the necessary $w_l(n)$, h_m, $x(n-l)$, and $f(n-l)$ for the algorithm at each step. For typical choices of the controller and plant filter lengths, the complexity and memory requirements of this algorithm are reasonable. As will be shown later, however, such is not the case for the natural extension of this algorithm to the multichannel control task.

III. AN ALTERNATE IMPLEMENTATION OF THE FILTERED-X LMS ALGORITHM

We now describe a new implementation of the single-channel filtered–X LMS algorithm [17]. This method combines the recently-introduced adjoint LMS algorithm of Wan [16] with a recently-developed method for delay compensation that first appeared in fast projection adaptive filters [18, 19]. To derive the implementation, we can write the coefficient updates of the original version of the algorithm in the form

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)\sum_{m=1}^{M} h_m x(n-l-m). \tag{5}$$

Define $\varepsilon_m(n)$ for $1 \le m \le M$ as

$$\varepsilon_m(n) = h_m \mu(n)\epsilon(n). \tag{6}$$

Then, (5) becomes

$$w_l(n+1) = w_l(n) - \sum_{m=1}^{M} \varepsilon_m(n)x(n-l-m). \tag{7}$$

We can represent the relation in (7) for M successive time steps as

$$w_l(n+1) = w_l(n-M+1) - \sum_{p=0}^{M-1} \sum_{m=1}^{M} \varepsilon_m(n-p)x(n-l-m-p). \tag{8}$$

We can expand the summation on the right-hand-side of (8) in a particularly useful way as

$$\begin{aligned} w_l(n+1) = \widehat{w}_l(n) & \\ - \varepsilon_1(n)x(n-l-1) & - \varepsilon_2(n)x(n-l-2) \cdots - \varepsilon_M(n)x(n-l-M) \\ & - \varepsilon_1(n-1)x(n-l-2) \cdots - \varepsilon_{M-1}(n-1)x(n-l-M) \\ & \qquad \ddots \qquad \vdots \\ & \qquad\qquad\qquad - \varepsilon_1(n-M+1)x(n-l-M) \end{aligned} \tag{9}$$

where we define the lth auxiliary coefficient $\widehat{w}_l(n)$ as

$$\widehat{w}_l(n) = w_l(n-M+1) - \sum_{p=1}^{M-1} \sum_{m=p+1}^{M} \varepsilon_m(n-M+m-p)x(n-l-M-p). \tag{10}$$

The expression in (9) indicates an important fact about the structure of the filtered–X LMS updates: the same input signals $x(n-l-m)$ are used in successive time instants to update the same coefficient $w_l(n)$. Such a result is readily apparent from (9), as each column of update terms on the right-hand-side of (9) depends on a single input signal value. We can exploit this structure to develop a set of coefficient updates that are grouped according to the individual $x(n-l-m)$ values appearing on the right-hand-side of (9). Such a scheme updates the lth auxiliary coefficient $\widehat{w}_l(n)$ rather than the actual controller coefficient $w_l(n)$. Define $e_m(n)$ as the sum of the error terms premultiplying the columns of $x(n-l-m)$ on the right-hand-side of (9) as

$$e_m(n) = \sum_{p=1}^{m} \varepsilon_p(n-m+p). \tag{11}$$

Then, it is straightforward to show that $\widehat{w}_l(n)$ can be updated as

$$\widehat{w}_l(n+1) = \widehat{w}_l(n) - e_M(n)x(n-M-l). \tag{12}$$

Thus, $\widehat{w}_l(n+1)$ is obtained by subtracting from $\widehat{w}_l(n)$ the last column of update terms on the right-hand-side of (9).

Note that $e_M(n)$ is a signal obtained by filtering $\mu(n)\epsilon(n)$ by the time-reversed plant impulse response $\{h_M, h_{M-1}, \ldots, h_1\}$, such that the coefficient update for $\widehat{w}_l(n)$ is the single-channel version of the adjoint LMS algorithm described by Wan [16]. What is novel is the relationship in (9), which provides the link between $\widehat{w}_l(n)$ and $w_l(n)$, or, equivalently, the link between the adjoint LMS and filtered–X LMS algorithms. We can use this expression to determine a method for computing $y(n)$ for the filtered–X LMS algorithm using the $\widehat{w}_l(n)$ obtained from the adjoint LMS algorithm. To proceed, we substitute the expression for $w_l(n+1)$ in (7) into (9). Using (11) to simplify the result, we obtain

$$w_l(n) = \widehat{w}_l(n) - \sum_{m=1}^{M-1} e_m(n-1)x(n-l-m-1). \tag{13}$$

We now consider the controller output calculation in (1). Substituting the expression for $w_l(n)$ in (13) into (1), we produce the equivalent expression

$$y(n) = \sum_{l=0}^{L-1} \widehat{w}_l(n)x(n-l) - \sum_{l=0}^{L-1} \sum_{m=1}^{M-1} e_m(n-1)x(n-l-m-1)x(n-l). \tag{14}$$

Equation	*MACs*
for $m = 1$ to $M-1$ do	
$\quad r_m(n) = r_m(n-1) + x(n)x(n-m-1) - x(n-L)x(n-L-m-1)$	$2M-2$
end	
$y(n) = \sum_{l=0}^{L-1} \widehat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_m(n)$	$L+M$
$\epsilon_\mu(n) = \mu(n)\epsilon(n)$	1
$e_1(n) = h_1\epsilon_\mu(n)$	1
for $m = 1$ to $M-1$ do	
$\quad e_{m+1}(n) = e_m(n-1) + h_{m+1}\epsilon_\mu(n)$	$M-1$
end	
for $l = 0$ to $L-1$ do	
$\quad \widehat{w}_l(n+1) = \widehat{w}_l(n) - e_M(n)x(n-M-l)$	L
end	
Total: $2L+4M-1$	

Table 1: An alternative filtered–X LMS algorithm.

Define the correlation term $r_m(n)$ as

$$r_m(n) = \sum_{l=0}^{L-1} x(n-l-m-1)x(n-l). \tag{15}$$

Then, (14) becomes

$$y(n) = \sum_{l=0}^{L-1} \widehat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_m(n). \tag{16}$$

Such a calculation is of reasonable complexity because of the sliding-window nature of the correlation term $r_m(n)$, which allows it to be recursively updated as

$$r_m(n) = r_m(n-1) + x(n)x(n-m-1) - x(n-L)x(n-L-m-1). \tag{17}$$

Moreover, $e_m(n)$ has a simple order-recursive update of the form

$$e_m(n) = \begin{cases} h_1\epsilon_\mu(n) & \text{if } m = 1 \\ e_{m-1}(n-1) + h_m\epsilon_\mu(n) & \text{if } 2 \le m \le M \end{cases}, \tag{18}$$

where

$$\epsilon_\mu(n) = \mu(n)\epsilon(n). \tag{19}$$

Collecting (12), (16), (17), (18), and (19), we obtain a consistent set of equations that exactly computes the output signal of the filtered–X LMS adaptive controller. Table 1 lists the operations for this new version of the algorithm as well as the number of multiply/accumulates (MACs) needed to implement each operation at each iteration. Since the total number of MACs is $2L+4M-1$, this algorithm is more computationally-complex than the original implementation of the filtered–X LMS algorithm, which only requires $2L+M+1$ MACs per iteration. However, in the multichannel case, the alternative implementation can save operations and memory storage, as we shall soon show.

IV. THE MULTICHANNEL FILTERED-X LMS ALGORITHM

We now describe the multichannel version of the filtered–X LMS algorithm in its original implementation [7, 8]. In multichannel control, I input sensors are used to collect I input signals $x^{(i)}(n)$, $1 \le i \le I$. The controller computes J output signals $y^{(j)}(n)$, $1 \le j \le J$ as

$$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} w_l^{(i,j)}(n) x^{(i)}(n-l), \tag{20}$$

where $w_l^{(i,j)}(n)$, $0 \le l \le L-1$ are the L FIR filter coefficients for the ith-input-to-jth-output channel of the controller. The J controller output signals propagate to the desired quiet region, where K error sensors measure the error signals $\epsilon^{(k)}(n)$, $1 \le k \le K$ as

$$\epsilon^{(k)}(n) = d^{(k)}(n) + \sum_{j=1}^{J} \sum_{m=-\infty}^{\infty} h_m^{(j,k)} y^{(j)}(n-m), \tag{21}$$

and $h_m^{(j,k)}$, $-\infty < m < \infty$ is the impulse response of the jth-output-to-kth-error channel of the acoustic plant.

To update $w_l^{(i,j)}(n)$ in the original filtered–X LMS algorithm, IJK filtered input signals $f^{(i,j,k)}(n)$ are computed as

$$f^{(i,j,k)}(n) = \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-m), \tag{22}$$

from which $w_l^{(i,j)}(n)$ is updated as

$$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) f^{(i,j,k)}(n-l), \tag{23}$$

where

$$\epsilon_\mu^{(k)}(n) = \mu(n) \epsilon^{(k)}(n). \tag{24}$$

A careful study of the filtered–X LMS algorithm described by (20), (22), (23), and (24) reveals the fact that this implementation requires $IJK(L+M)+K$ MACs to compute the coefficient updates, even though computing the controller outputs only requires IJL MACs. Thus, the complexity of the update calculations is more than K times the complexity of the input-output calculations. For systems with a large number of error sensors, the computational burden of the coefficient updates can overwhelm the capabilities of the processor chosen for the control task.

The standard implementation of the filtered–X LMS algorithm also has memory requirements that can exceed the capabilities of a chosen processor. The total storage needed is $IJ(K+1)L + JKM + I\max(L, M+1) + K$, and for long controller filter lengths, the bulk of this storage is for the $IJKL$ filtered input signals $f^{(i,j,k)}(n-l)$. Clearly, it is desirable to find alternative implementations of the filtered–X LMS algorithm that have reduced computational and memory requirements. We now present one such algorithm that is based on the method described in Section III.

V. A MULTICHANNEL FILTERED-X LMS ALGORITHM WITH REDUCED COMPLEXITY

We consider the multichannel extension of the new version of the filtered–X LMS algorithm in Section III. To determine the appropriate grouping of terms for the updates, we substitute the expression for $f^{(i,j,k)}(n-l)$ in (22) into the update in (23) to get

$$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-l-m) \quad (25)$$

$$= w_l^{(i,j)}(n) - \sum_{m=1}^{M} \varepsilon_m^{(j)}(n) x^{(i)}(n-l-m), \quad (26)$$

where we have defined the JM terms $\varepsilon_m^{(j)}(n)$ for $1 \le j \le J$ and $1 \le m \le M$ as

$$\varepsilon_m^{(j)}(n) = \sum_{k=1}^{K} h_m^{(j,k)} \epsilon_\mu^{(k)}(n). \quad (27)$$

Comparing (26) with (7), they are seen to be of a similar form. Thus, we can use a method analogous to that in Section III to implement the multichannel filtered–X LMS algorithm. For this implementation, we define the JM terms $e_m^{(j)}(n)$ as

$$e_m^{(j)}(n) = \sum_{p=1}^{m} \varepsilon_p^{(j)}(n-m+p). \quad (28)$$

Then, we can define a set of IJL auxiliary coefficients $\widehat{w}_l^{(i,j)}(n)$ whose updates are given by

$$\widehat{w}_l^{(i,j)}(n+1) = \widehat{w}_l^{(i,j)}(n) - e_M^{(j)}(n) x^{(i)}(n-M-l). \quad (29)$$

To compute the controller outputs, the multichannel equivalent of (16) is

$$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \widehat{w}_l^{(i,j)}(n) x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1) r_m(n), \quad (30)$$

where $r_m(n)$ in this case is defined as

$$r_m(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} x^{(i)}(n-l) x^{(i)}(n-l-m-1). \quad (31)$$

In analogy with (17), $r_m(n)$ can be recursively computed as

$$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left(x^{(i)}(n) x^{(i)}(n-m-1) - x^{(i)}(n-L) x^{(i)}(n-L-m-1) \right) \quad (32)$$

to minimize the number of operations per time instant. Similarly, $e_m^{(j)}(n)$ has an update similar to that in (18) as given by

$$e_m^{(j)}(n) = \begin{cases} \sum_{k=1}^{K} h_1^{(j,k)} \epsilon_\mu^{(k)}(n) & \text{if } m = 1 \\ e_{m-1}^{(j)}(n-1) + \sum_{k=1}^{K} h_m^{(j,k)} \epsilon_\mu^{(k)}(n) & \text{if } 2 \le m \le M \end{cases} \quad (33)$$

Equation	*MACs*
for $m = 1$ to $M-1$ do	
$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left(x^{(i)}(n)x^{(i)}(n-m-1) - x^{(i)}(n-L)x^{(i)}(n-L-m-1) \right)$	$2I(M-1)$
end	
for $j = 1$ to J do	
$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_m(n)$	$IJL + J(M-1)$
end	
for $k = 1$ to K do	
$\epsilon_\mu^{(k)}(n) = \mu(n)\epsilon^{(k)}(n)$	K
end	
for $j = 1$ to J do	
$e_1^{(j)}(n) = \sum_{k=1}^{K} h_1^{(j,k)} \epsilon_\mu^{(k)}(n)$	JK
for $m = 1$ to $M-1$ do	
$e_{m+1}^{(j)}(n) = e_m^{(j)}(n-1) + \sum_{k=1}^{K} h_{m+1}^{(j,k)} \epsilon_\mu^{(k)}(n)$	$JK(M-1)$
end	
for $i = 1$ to I do	
for $l = 0$ to $L-1$ do	
$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l)$	IJL
end	
end	
end	
Total: $2IJL + JKM + (2I+J)(M-1) + K$	

Table 2: A multichannel filtered–X LMS algorithm with reduced complexity.

Collecting (24), (29), (30), (32), and (33), we obtain an alternative, equivalent implementation of the multichannel filtered–X LMS algorithm. Table 2 lists the operations of this implementation, along with the number of MACs required to implement each operation. The algorithm employs $2IJL + JKM + (2I+J)(M-1) + K$ MACs per iteration, and it requires $IJL + JKM + IL + (I+J+1)M + K - 1$ memory locations to implement.

VI. COMPLEXITY COMPARISONS

We now compare the computational and memory requirements of the original and new implementations of the multichannel filtered–X LMS algorithms. In this comparison, we consider systems with different numbers of input sensors, output actuators, and error sensors under three different problem scenarios. Each scenario is defined by specific choices of the controller filter length L and plant model filter length M that would be appropriate for a particular type of noise or vibration control task. In each case, we present the quantities $R_F^{(C)}$ and $R_F^{(M)}$ that denote the ratios of the numbers of MACs and memory locations, respectively, required by the new implementation of the multichannel filtered–X LMS algorithm with respect to the numbers of MACs and memory locations needed for the algorithm's original implementation. For comparison, we also provide the ratios $R_A^{(C)}$ and $R_A^{(M)}$ for the multichannel adjoint LMS algorithm [16] with respect to the original multichannel filtered–X LMS algorithm. Since the adjoint LMS algorithm equations are used within

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
1	2	2	0.7512	0.9900	0.7235	0.7765	4	6	4	0.3574	0.3974	0.3313	0.3348
1	3	2	0.7508	0.9502	0.6933	0.7301	2	6	6	0.3506	0.3906	0.3469	0.3515
1	4	2	0.7506	0.9302	0.6772	0.7054	2	8	6	0.3505	0.3865	0.3412	0.3446
1	6	2	0.7504	0.9101	0.6605	0.6797	2	8	8	0.3082	0.3359	0.3096	0.3123
2	2	2	0.6259	0.8055	0.6259	0.6559	4	8	8	0.2311	0.2495	0.2288	0.2303
2	3	2	0.6256	.0.7654	0.5877	0.6085	8	8	8	0.1925	0.2063	0.1820	0.1828
2	4	2	0.6255	0.7453	0.5672	0.5832	4	8	12	0.1845	0.1972	0.1927	0.1937
2	6	2	0.6253	0.7252	0.5459	0.5568	4	12	12	0.1844	0.1949	0.1888	0.1895
1	4	4	0.5726	0.6752	0.5358	0.5523	4	16	12	0.1844	0.1938	0.1869	0.1874
1	6	4	0.5722	0.6635	0.5241	0.5353	8	8	12	0.1449	0.1544	0.1444	0.1449
2	3	3	0.5009	0.6025	0.4928	0.5085	8	8	16	0.1202	0.1274	0.1244	0.1248
1	4	6	0.5015	0.5733	0.4771	0.4888	8	12	16	0.1201	0.1257	0.1211	0.1214
3	3	3	0.4552	0.5424	0.4490	0.4601	8	16	16	0.1201	0.1249	0.1195	0.1197
3	6	3	0.4549	0.5130	0.4111	0.4168	8	32	16	0.1200	0.1236	0.1170	0.1171
2	4	4	0.4294	0.4979	0.4209	0.4305	16	16	16	0.1000	0.1036	0.0926	0.0927
2	6	4	0.4291	0.4862	0.4060	0.4125	16	32	16	0.1000	0.1024	0.0901	0.0901
2	8	4	0.4290	0.4804	0.3985	0.4033	8	16	32	0.0817	0.0842	0.0901	0.0903
4	4	4	0.3576	0.4090	0.3484	0.3536	16	16	32	0.0613	0.0631	0.0625	0.0625
2	4	6	0.3510	0.3989	0.3582	0.3651	32	32	32	0.0510	0.0520	0.0466	0.0466

Table 3: Complexity comparison, $L = 50$, $M = 25$.

the new filtered–X LMS algorithm implementation, we have that $R_A^{(C)} < R_F^{(C)}$ and $R_A^{(M)} < R_F^{(M)}$, although the requirements of these two algorithms are similar for systems with a large number of channels.

The first situation considered is a broadband noise control task in which the controller and plant model filter lengths are $L = 50$ and $M = 25$. These choices are typical for this type of control problem and offer a reasonable balance between the performance and robustness of the controller in this situation for fixed hardware resources. Table 3 shows the complexity and memory ratios for the different cases considered. As can be seen for all of the cases considered, the number of multiplies required for the new implementation of the multichannel filtered–X LMS algorithm is less than that of the original algorithm, and this difference is significant for systems with a large number of channels. In fact, for an N-input, N-output, N-error system, the complexity of the new implementation is approximately 80% of the original implementation when $N = 2$, 40% of the original when $N = 4$, 20% of the original when $N = 8$, and 10% of the original when $N = 16$. In addition, the number of memory locations required by the new implementation is also reduced and is less than 10% of the original algorithm's memory requirements for $I = J = K = N = 16$. These savings are significant, as they allow a multichannel control system to be implemented on a much-simpler hardware platform.

We now consider a narrowband control task in which $L = 2$ and $M = 10$. Such a situation is typical of narrowband noise control problems in which each input signal is a single sinusoid of a different frequency; thus, each channel of the controller is dedicated to one tonal component of the unwanted noise or vibration acoustic field. Table 4 lists the ratio of MACs and memory locations for the two algorithms with respect to the original filtered–X LMS algorithm in this situation. As can be seen, except for systems with a small number of channels, the new implementation of the filtered–X LMS algorithm requires only a fraction of the MACs and memory locations used by the

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5472	1.0566	1.0682	1.1705	6	12	6	0.1901	0.2306	0.0.5900	0.5950
2	3	2	0.5443	0.9430	1.0667	1.1417	6	6	12	0.1663	0.1970	0.5398	0.5450
2	4	2	0.5429	0.8857	1.0658	1.1250	6	12	12	0.1653	0.1859	0.5274	0.5301
2	2	4	0.4902	0.7549	0.9315	0.9932	8	16	12	0.1307	0.1461	0.4570	0.4587
2	4	4	0.4851	0.6634	0.9173	0.9511	8	12	16	0.1245	0.1380	0.4433	0.4450
2	6	4	0.4834	0.6325	0.9119	0.9352	8	16	16	0.1243	0.1359	0.4401	0.4414
2	6	6	0.4631	0.5638	0.8525	0.8687	8	16	24	0.1178	0.1256	0.4227	0.4236
2	6	8	0.4527	0.5287	0.8209	0.8333	8	16	32	0.1146	0.1204	0.4138	0.4145
4	6	4	0.2824	0.3870	0.7576	0.7746	16	32	16	0.0723	0.0781	0.2937	0.2941
4	8	6	0.2586	0.3193	0.6933	0.7025	16	16	32	0.0625	0.0669	0.2707	0.2711
4	8	8	0.2468	0.2926	0.6625	0.6696	16	32	32	0.0623	0.0652	0.2669	0.2671

Table 4: Complexity comparison, $L = 2$, $M = 10$.

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5745	0.9787	0.9098	0.9877	6	12	6	0.2109	0.2442	0.3911	0.3940
2	3	2	0.5735	0.8886	0.8779	0.9331	6	6	12	0.1629	0.1886	0.3375	0.3405
2	4	2	0.5730	0.8434	0.8604	0.9032	6	12	12	0.1625	0.1796	0.3257	0.3273
2	2	4	0.4656	0.6832	0.7488	0.7956	8	16	12	0.1354	0.1482	0.2755	0.2764
2	4	4	0.4636	0.6092	0.7102	0.7350	8	12	16	0.1227	0.1341	0.2603	0.2613
2	6	4	0.4629	0.5844	0.6963	0.7131	8	16	16	0.1227	0.1324	0.2578	0.2585
2	6	6	0.4226	0.5057	0.6381	0.6499	8	16	24	0.1098	0.1163	0.2394	0.2398
2	6	8	0.4016	0.4648	0.6067	0.6158	8	16	32	0.1033	0.1082	0.2299	0.2303
4	6	4	0.3086	0.3937	0.5452	0.5560	16	32	16	0.0817	0.0865	0.1690	0.1692
4	8	6	0.2639	0.3138	0.4760	0.4818	16	16	32	0.0620	0.0656	0.1434	0.1436
4	8	8	0.2408	0.2787	0.4440	0.4485	16	32	32	0.0619	0.0644	0.1409	0.1410

Table 5: Complexity comparison, $L = 10$, $M = 20$.

original implementation, and this fraction is reduced as the number of channels is increased. Thus, the new algorithm implementation reduces the controller's hardware complexity in narrowband control situations as well.

The third problem scenario considered is a feedforward control task in which $L = 10$ and $M = 20$. These choices are typical for noise and vibration control tasks in which the input signals are measured by physical sensors, but the primary goal of the controller is to attenuate a relatively few number of tonal components in the desired quiet region. Table 5 lists the respective complexity and memory usage ratios for the different algorithms for different cases. As in the previous cases, we find that the new implementation of the filtered–X LMS algorithm save computations and memory locations, and the savings is significant for systems with a large number of channels.

VII. NUMERICAL ISSUES

In this section, we briefly consider the effects that numerical errors due to finite precision calculations have on the performance of the new implementation of the filtered–X LMS algorithm. One

important feature of the LMS algorithm in adaptive filtering is its robust behavior in the presence of various approximations and errors that are often introduced in a real-world implementation. The original implementation of the filtered–X LMS algorithm also shares similar robustness properties and is known to perform adequately despite any numerical errors introduced within the controller calculations and any delays introduced by the plant. Moreover, since the adjoint LMS algorithm [16] is one form of a filtered-error LMS algorithm [15], it too shares many of the robust convergence properties of other stochastic gradient algorithms. Since the the new implementation of the filtered–X LMS algorithm applies a particular form of delay compensation to the adjoint LMS algorithm, an issue remains as to the numerical properties of the delay compensation technique, particularly as they affect the long-term performance of the overall system.

While an analysis of the numerical properties of the delay compensation technique used in our new algorithm implementation are beyond the scope of this paper, it appears from simulation that the compensation method does not significantly alter the robust numerical properties of the filtered–X LMS algorithm in this implementation. The only possible source of numerical difficulties is the method for calculating $r_m(n)$ in (32), as this update is marginally-stable due to its recursive nature. Thus, numerical errors in $r_m(n)$ can grow linearly over time in a finite-precision environment. Fortunately, the growth in these errors can be easily prevented using several well-known procedures. Perhaps the simplest procedure is to periodically recalculate $r_m(n)$ using (32), a procedure that requires extra additions and memory locations but does not require additional multiplications. Another solution is to introduce a leakage factor into the calculation of $r_m(n)$. One particularly-useful method, described in the single-channel case for simplicity, is given by

$$r_m(n) = \begin{cases} \lambda r_m(n-1) + x(n)x(n-m-1) - \lambda x(n-L)x(n-L-m-1) & \text{if } n \bmod L = 0 \\ r_m(n-1) + x(n)x(n-m-1) - \lambda x(n-L)x(n-L-m-1) & \text{otherwise,} \end{cases} \quad (34)$$

where λ is slightly less than one. This method alters the value of $r_m(n)$ slightly, but for values of λ close to one, the errors introduced into the calculations for $y(n)$ do not significantly affect the overall behavior of the control system. Moreover, the update in (34) adds little complexity to the overall system.

Figure 2 plots the error signal $\epsilon^{(1)}(n)$ of three different four-input, three-output, four-error active noise control systems as applied to air conditioner data taken in an anechoic environment [20]. In this case, all calculations were performed in the MATLAB floating-point environment. Shown for comparison are the performances of the original filtered–X LMS algorithm, the new filtered–X LMS algorithm of Table 2, and the new filtered–X LMS algorithm with the multichannel version of (34) used in place of (32), where $\lambda = 1 - 2^{-10}$, $\mu(n) = 0.5$, $L = 500$, and $M = 200$. As can be seen, the three convergence curves are indistinguishable on this plot, indicating that the numerical stabilization provided by (34) does not significantly alter the performance of the overall system.

VIII. CONCLUSIONS

We have described a new implementation of the multichannel filtered–X LMS algorithm for feedforward active noise and vibration control tasks. This implementation guarantees the same input-output behavior of the original implementation while requiring only a fraction of the computational effort and memory required for the original algorithm. Because of the pervasiveness of the filtered–X LMS algorithm in active noise and vibration control systems, the new implementation is expected to have a significant impact on the practicality and cost of these schemes in real-world applications.

ACKNOWLEDGEMENT

This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy

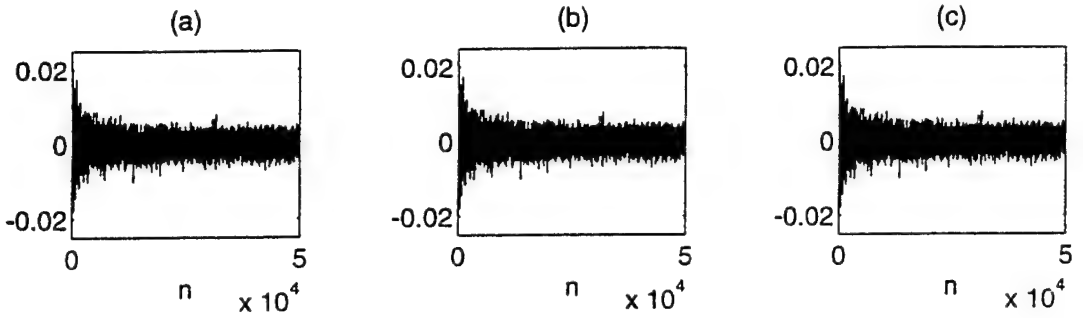

Figure 2: Error signals $\epsilon^{(1)}(n)$ for the multichannel filtered–X LMS algorithms as applied to air conditioner compressor data: (a) original implementation, (b) new implementation, and (c) new implementation with leakage update for $r_m(n)$.

of the federal government, and no official endorsement should be inferred.

Data was provided by SRI International, Menlo Park, CA.

REFERENCES

[1] "Active attenuation of noise–the state of the art," G.E. Warnaka, *Noise Contr. Eng.*, **18**, 100-110 (1982).

[2] "Recent advances in active noise control," J.C. Stevens and K.K. Ahuja, *AIAA J.*, **29**, 1058-1067 (1991).

[3] "Active noise control," S.J. Elliott and P.A. Nelson, *IEEE Sig. Proc. Mag.*, **10**, 12-35 (1993).

[4] "Active control of sound and vibration," C.R. Fuller and A.H. von Flotow, *IEEE Contr. Sys. Mag.*, **15**, 9-19 (1995).

[5] "Perspectives on active noise and vibration control," E.F. Berkman and E.K. Bender, *Sound Vib.*, **31**, 80-94 (1997).

[6] M.O. Tokhi and R.R. Leitch, *Active Noise Control* (Clarendon, Oxford, 1992).

[7] P.A. Nelson and S.J. Elliott, *Active Control of Sound* (Academic Press, London, 1992).

[8] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (Wiley, New York, 1996).

[9] C.R. Fuller, S.J. Elliott, and P.A. Nelson, *Active Control of Vibration* (Academic Press, London, 1996).

[10] "Process of silencing sound oscillations," P. Leug, U.S. Patent 2,043,416 (1936).

[11] B. Widrow and S.D. Stearns, *Adaptive Signal Processing* (Prentice-Hall, Englewood Cliffs, 1985).

[12] S. Haykin, *Adaptive Filter Theory*, 3rd ed. (Prentice-Hall, Englewood Cliffs, 1995).

[13] "Time- and frequency-domain X-block least-mean-square algorithms for active noise control," Q. Shen and A.S. Spanias, *Noise Contr. Eng. J.*, **44**, 281-293 (1996).

[14] "Adaptive filters employing partial updates," S.C. Douglas, *IEEE Trans. Circ. Syst. II: Anal. Dig. Sig. Proc.*, **44**, 209-216 (1997)

[15] B. Widrow and E. Walach, *Adaptive Inverse Control* (Prentice-Hall, Upper Saddle River, 1996).

[16] "Adjoint LMS: An efficient alternative to the filtered–X LMS and multiple error LMS algorithms," E.A. Wan, *Proc. ICASSP-96*, Atlanta, GA, 1842-1845 (1996).

[17] "Fast, exact filtered–X LMS and LMS algorithms for multichannel active noise control," S.C. Douglas, *Proc. ICASSP-97*, Munich, Germany, 399-402 (1997).

[18] "The fast affine projection algorithm," S. Gay and S. Tavathia, *Proc. ICASSP-96*, Detroit, MI, 3023-3026 (1995).

[19] "Fast projection algorithm and its step size control," M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, *Proc. ICASSP-96*, Detroit, MI, 945-948 (1995).

[20] "Active control of complex noise problems using a broadband multichannel controller," D.K. Peterson, W.A. Weeks, and W.C. Nowlin, *Proc. NOISE-CON*, Ft. Lauderdale, FL, 315 (1994).

Numerically-Stable Implementations of the Sliding DFT, Sliding DCT, and Sliding-Window RLS Algorithms *

Scott C. Douglas and Jihee K. Soh †
Department of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112

Abstract– **The sliding discrete Fourier transform (DFT) is an important tool for data analysis and adaptive filtering tasks, as is the simpler task of calculating the sum of L signal values over a sliding window. While such systems can be recursively implemented, numerical errors accumulate in the resulting outputs. This paper presents a periodically-time-varying recursive system whose output at every Lth sample time is equivalent to that of the sliding DFT and whose numerical errors do not grow with time. Moreover, the mathematical difference between the system's output and that of the non-recursive DFT can be made arbitrarily small with the proper choice of the leakage parameter. We provide a statistical analysis of the numerical performance of the proposed technique, showing that it outperforms a similar approximate method employing an exponentially-decaying window. When applied to sliding-window-covariance recursive least-squares adaptive filters, the proposed technique mitigates this system's error accumulation without any significant increases in the overall computational complexity. An extension of the method to the sliding discrete cosine transform (DCT) is also provided.**

submitted to

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II:

Analog and Digital Signal Processing

---

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

†Please address correspondence to: Scott C. Douglas, Department of Electrical Engineering, University of Utah, 50 South Central Campus Drive, Room 3280, Salt Lake City, UT 84112. (801) 581-4445. FAX: (801) 581-5281. Electronic mail address: douglas@ee.utah.edu. World Wide Web URL: http://www.elen.utah.edu/~douglas.

1 Introduction

Consider the following discrete-time linear time-invariant system:

$$y(k) = \sum_{m=0}^{L-1} e^{-j2\pi ml/L} x(k-m), \tag{1}$$

where l is an integer in the range $0 \leq l \leq (L-1)$. This system computes the lth bin value of the sliding discrete Fourier transform (DFT) of the signal $x(k)$. Such a system is useful in spectral estimation [1] as well as in transform-domain adaptive filtering [2].

When $l = 0$, (1) computes the running sample average of a signal across an L-element window. This ubiquitous system is useful for numerous signal processing tasks. In particular, it appears in certain delay-compensation techniques that are useful for fast affine projection (FAP) adaptive filtering [3, 4], for fast exact least-mean-square (LMS) adaptive filters [5], for pipelined implementations of the LMS adaptive filter [6, 7], and for adaptive feedfoward control schemes in active noise control [8, 9, 10]. In addition, if $x(k-m)$ represents the squared estimation error of an N-coefficient FIR adaptive filter, than $y(k)$ is the sum-of-squared-errors across a sliding data window. In this case, the recursive minimization of $y(k)$ over time yields different forms of the sliding-window-covariance recursive least-squares (SWC-RLS) adaptive filter [11, 12]. Such a system also plays an important role in the FAP algorithm [3, 4].

It is well-known that the system in (1) can be implemented recursively as

$$y(k) = e^{-j2\pi l/L} y(k-1) - x(k-L) + x(k). \tag{2}$$

Such an implementation is quite useful, as it only requires a single (complex) multiply and two additions per time instant. However, this recursive implementation is marginally-stable due to the pole-zero cancellation that occurs at $z = e^{-j2\pi l/L}$ in the transfer function of this linear time-invariant system. Thus, finite-precision errors will linearly accumulate in the value of $y(k)$ over time. If left unchecked, such errors can make (2) useless for signal processing purposes. For this reason, the following approximate system has been proposed [2]:

$$\widehat{y}(k) = \sum_{m=0}^{L-1} (\widehat{\lambda} e^{-j2\pi l/L})^m x(k-m), \tag{3}$$

where $\widehat{\lambda}$ is a constant that is slightly less than one. The recursive implementation of this system is

$$\widehat{y}(k) = \left(\widehat{\lambda} e^{-j2\pi l/L}\right) \widehat{y}(k-1) - \widehat{\lambda}^L x(k-L) + x(k), \tag{4}$$

which requires two multiplies and two additions per time instant. While this system is numerically-stable, it no longer computes the exact value of $y(k)$ in (1), although the difference between $y(k)$ and $\widehat{y}(k)$ can be made small via suitable choice of $\widehat{\lambda}$. An alternative solution is to periodically restart the calculation of $y(k)$. This method does not require any additional multiplies, but it does require more additions and memory. Moreover, the two-tiered coding strategy needed for the implementation of the system's restart may be undesirable, particularly in tasks such as SWC-RLS adaptive filtering in which a periodic restart can incur a significant computational cost [3].

When implementing the sliding DFT, two other methods are of particular interest. The LMS spectrum analyzer uses a variant of the LMS algorithm to compute an L-point sliding DFT [13]. While its complexity is similar to that in (2), simulations in Section 4 show that it suffers from similar numerical accumulation effects when implemented in a floating-point environment, although its numerical properties in a fixed-point environment appear to be better than those of (2) [14]. When L is a power of two, the sliding FFT is a non-recursive method for computing an L-point sliding DFT that requires $O(L)$ multiplies per time instant to implement [15, 16]. This system requires $O(L \log_2 L)$ memory locations, however, which can be prohibitive for some applications.

In this paper, we present a technique that, like (4), approximates the system in (1) using a simple recursive update. Our technique is a periodically-time-varying system designed so that any numerical errors that are introduced by finite-precision arithmetic at any time instant exponentially decay to zero over time. Like the approximate method in (4), the new method requires two multiplications and two additions at every time instant and is simple to implement. However, unlike (4) in which $\widehat{y}(k) \neq y(k)$ in general, our technique produces an output signal that is mathematically-equivalent to $y(k)$ in (1) at every Lth time instant. At other time instants, the difference between the output of the proposed system and $y(k)$ in (1) can be made arbitrarily small through the proper choice of the leakage parameter $\overline{\lambda}$. Moreover, the new technique does not duplicate the memory requirements of the system, as does a periodic restart.

The organization of this paper is as follows. In the next section, we introduce the proposed technique. An analysis of the new method's statistical performance when $l = 0$ is given in Section 3, where it is shown that its average bias and squared error are lower than that of the estimator in (4) for equivalent levels of numerical precision error. In Section 4, applications of the new technique to both transform-domain adaptive filters and SWC-RLS adaptive filters are provided, showing that the new method stabilizes the marginal instabilities in these systems with little to no additional

computational cost. Section 5 provides a simple extension of the technique to the sliding discrete cosine transform (DCT). Section 6 presents our conclusions.

2 A Numerically-Stable Recursive Sliding-Window Estimator

The proposed method for approximating (1) is a periodically-time-varying linear system. If numerical errors are not present, the system input-output relation is equivalent to the system

$$\overline{y}(k) = \sum_{m=0}^{L-1} h_l(k,m)x(k-m) \tag{5}$$

$$h_l(k,m) = \begin{cases} e^{-j2\pi ml/L} & \text{if } k\text{mod}L \geq m \\ \overline{\lambda}e^{-j2\pi ml/L} & \text{if } k\text{mod}L < m \end{cases}, \tag{6}$$

where $x(k)$ and $\overline{y}(k)$ are the input and output signals of the system, respectively, $h_l(k,m)$ is the time-varying impulse response of the system and $\overline{\lambda}$ is slightly less than one. Thus, when $k\text{mod}L = L-1$, the value of $\overline{y}(k)$ is exactly the same as $y(k)$ in (1). At other sample instants, the output of the system in (5)–(6) deviates from that in (1). A simple calculation shows that the magnitude of this deviation is no greater than

$$|\overline{y}(k) - y(k)| \leq (1-\overline{\lambda}) \sum_{m=1}^{L-1} |x(k-m)|, \tag{7}$$

such that by proper selection of the value of $\overline{\lambda}$, the mathematical difference between these two estimators can be made arbitrarily small.

The system in (6) has a simple recursive implementation that makes it a useful candidate for approximating the sliding DFT:

$$\overline{y}(k) = \begin{cases} \overline{\lambda}[e^{-j2\pi l/L}\overline{y}(k) - x(k-L)] + x(k), & \text{if } k\text{mod}L = 0 \\ e^{-j2\pi l/L}\overline{y}(k) - \overline{\lambda}x(k-L) + x(k), & \text{otherwise.} \end{cases} \tag{8}$$

This implementation requires two multiplications and two additions as well as the control logic required to determine when $k\text{mod}L = 0$. Note that many digital signal processors (DSPs) have built-in circular pointer indexing schemes that make testing for the condition $k\text{mod}L = 0$ simple to implement.

3 Performance Analysis

We now analyze the performances of the estimators in (4) and (8) in a statistical framework. For simplicity, we consider the case where $l = 0$ and $x(k)$ is real-valued, such that the outputs of

these estimators approximate the sum of the L most-recent samples of $x(k)$. In these analyses, we assume that $x(k)$ is a stationary uncorrelated random process with mean μ_x and variance σ_x^2. In each case, we determine three quantities:

- the average bias in the outputs of each system with respect to $y(k)$,
- the variance of the difference between the outputs of each system and $y(k)$, and
- the variance of the numerical error in the output of each system assuming a standard statistical model for the quantization errors.

3.1 Exponentially-Windowed Estimator

We first analyze the performance of the system in (4) under the above assumptions. These results provide a baseline with which the performance of (8) can be compared.

The average bias of this estimator assuming infinite precision calculations is

$$\widehat{\mu} = E\{y(k) - \widehat{y}(k)\} \tag{9}$$

$$= E\left\{\sum_{m=0}^{L-1}(1-\widehat{\lambda}^m)x(k-m)\right\}. \tag{10}$$

Under our statistical assumptions, this expression becomes

$$\widehat{\mu} = \sum_{m=1}^{L-1}(1-\widehat{\lambda}^m)\mu_x = \left(L-1-\frac{\widehat{\lambda}-\widehat{\lambda}^L}{1-\widehat{\lambda}}\right)\mu_x. \tag{11}$$

The variance of the estimation error $y(k) - \widehat{y}(k)$, denoted as $\widehat{\sigma}^2$, is given by

$$\widehat{\sigma}^2 = E\left\{[y(k) - \widehat{y}(k) - \widehat{\mu}]^2\right\} \tag{12}$$

$$= E\left\{\left[\sum_{m=0}^{L-1}(1-\widehat{\lambda}^m)(x(k-m)-\mu_x)\right]^2\right\}. \tag{13}$$

With our assumptions, the RHS of (13) is

$$\widehat{\sigma}^2 = \sum_{m=1}^{L-1}(1-\widehat{\lambda}^m)^2\sigma_x^2 \tag{14}$$

$$= \left(L-1-2\frac{\widehat{\lambda}-\widehat{\lambda}^L}{1-\widehat{\lambda}}+\frac{\widehat{\lambda}^2-\widehat{\lambda}^{2L}}{1-\widehat{\lambda}^2}\right)\sigma_x^2. \tag{15}$$

We now consider the performance of (4) in a finite-precision environment, in which $\underline{\widehat{y}}(k)$, the finite-precision equivalent of $\widehat{y}(k)$, is approximately

$$\underline{\widehat{y}}(k) = \widehat{\lambda}\underline{\widehat{y}}(k-1) - \widehat{\lambda}^L x(k-L) + x(k) + \epsilon_i(k) + \epsilon_o(k), \tag{16}$$

where $\epsilon_i(k)$ and $\epsilon_o(k)$ are the numerical errors associated with the first and second terms, respectively, on the RHS of (16). Defining $\widehat{\epsilon}(k)$ as

$$\widehat{\epsilon}(k) \;=\; \underline{\widehat{y}}(k) - \widehat{y}(k), \tag{17}$$

we can develop a recursive equation for $\widehat{\epsilon}(k)$ given by

$$\widehat{\epsilon}(k) \;=\; \widehat{\lambda}\widehat{\epsilon}(k-1) + \epsilon_i(k) + \epsilon_o(k). \tag{18}$$

Assuming that $\epsilon_i(k)$ and $\epsilon_o(k)$ are zero mean, independent, and identically-distributed (i.i.d.) with equal variances σ_ϵ^2, we obtain the limiting variance of $\widehat{\epsilon}(k)$ as

$$\lim_{k\to\infty} \sigma_{\widehat{\epsilon}}^2(k) \;=\; \frac{2\sigma_\epsilon^2}{1-\widehat{\lambda}^2}. \tag{19}$$

3.2 Periodically Time-Varying Estimator

We now analyze the statistical performance of (8) under our assumptions. Since the system has a time-varying impulse response, we consider quantities that are averaged over L sample times in order to provide a fair comparison with our previously-derived results.

We first consider the time-averaged bias, denoted as

$$\overline{\mu} \;=\; \frac{1}{L}\sum_{n=k-L+1}^{k} E\{y(n) - \overline{y}(n)\} \tag{20}$$

$$\;=\; \frac{1}{L}\sum_{n=k-L+1}^{k} E\left\{\sum_{m=0}^{L-1}(1-h_0(n,m))x(n-m)\right\}. \tag{21}$$

Employing the expression for $h_0(k,m)$ in (6) we can simply the above expression as

$$\overline{\mu} \;=\; \frac{1}{L}\sum_{n=k-L+1}^{k}\sum_{m=n \bmod L}^{L-1}(1-\overline{\lambda})E\{x(n-m)\} \tag{22}$$

$$\;=\; \frac{1}{L}\sum_{m=1}^{L-1}(1-\overline{\lambda})m\mu_x \;=\; \frac{L-1}{2}(1-\overline{\lambda})\mu_x. \tag{23}$$

The time-averaged error variance of the new estimator's output, denoted as $\overline{\sigma}^2$, is defined as

$$\overline{\sigma}^2 \;=\; \frac{1}{L}\sum_{n=k-L+1}^{k} E\left\{[y(n) - \overline{y}(n) - \overline{\mu}]^2\right\} \tag{24}$$

$$\;=\; \frac{1}{L}\sum_{n=k-L+1}^{k} E\left\{\left[\sum_{m=0}^{L-1}(1-h_0(n,m))(x(n-m)-\mu_x)\right]^2\right\}. \tag{25}$$

Using simplifications that are similar to those used in (22)–(23), the right-hand-side of (25) is

$$\overline{\sigma}^2 \;=\; \frac{1}{L}\sum_{m=1}^{L-1}(1-\overline{\lambda})^2 m\sigma_x^2 \;=\; \frac{L-1}{2}(1-\overline{\lambda})^2\sigma_x^2. \tag{26}$$

Finally, we consider the accumulation of errors in $\overline{y}(k)$ in a finite-precision environment. The numerical model in this case is similar to that in (16) and is given by

$$\underline{\overline{y}}(k) \;=\; \begin{cases} \overline{\lambda}[\underline{\overline{y}}(k)-x(k-L)]+x(k)+\epsilon_o(k)+\epsilon_i(k), & \text{if } k\text{mod}L=0 \\ \underline{\overline{y}}(k)-\overline{\lambda}x(k-L)+x(k)+\epsilon_i(k), & \text{otherwise.} \end{cases} \tag{27}$$

For simplicity, we only consider the worst-case performance by determining the level of numerical errors in $\underline{\overline{y}}(k)$ at time instants $k = nL-1$, where the value of $\underline{\overline{y}}(k)$ equals $y(k)$ in an infinite-precision environment. By defining $\overline{\epsilon}(nL-1) = \underline{\overline{y}}(nL-1) - y(nL-1)$, it is straightforward to show that

$$\overline{\epsilon}(nL-1) \;=\; \overline{\lambda}\overline{\epsilon}((n-1)L-1)+\epsilon_o((n-1)L)+\sum_{m=1}^{L}\epsilon_x(nL-m). \tag{28}$$

Thus, the limiting value of the variance of $\overline{\epsilon}(nL-1)$ is

$$\lim_{n\to\infty}\sigma_{\overline{\epsilon}}^2(nL-1) \;=\; \frac{(L+1)\sigma_\epsilon^2}{1-\overline{\lambda}^2}. \tag{29}$$

3.3 Performance Comparison

We now compare the performances of the recursive estimators in (4) and (8). For this comparison, we equate the numerical precision error variances as expressed by (19) and (29), respectively, and solve for $\widehat{\lambda}$ for the exponentially-windowed estimator in terms of $\overline{\lambda}$ for the new estimator. This operation yields the relationship

$$\widehat{\lambda} \;=\; \sqrt{1-\frac{2}{L+1}(1-\overline{\lambda}^2)}. \tag{30}$$

Then, we compute the average biases and error variances for each algorithm using their respective parameter values $\overline{\lambda}$ and $\widehat{\lambda}$.

Table 1 shows the values of the biases $\widehat{\mu}$ and $\overline{\mu}$ as well as the error variances $\widehat{\sigma}^2$ and $\overline{\sigma}^2$, normalized by $L\mu_x$ and $L\sigma_x^2$, respectively. In each case, it is observed that the proposed scheme has less bias and less error variance than the exponentially-windowed scheme. Thus, the proposed scheme is to be preferred over the existing stabilization technique from the standpoint of numerical performance. In addition, we have simulated the behaviors of both algorithms in the MATLAB data analysis environment, where sequences of zero mean uncorrelated Gaussian random variables

of unit variance were used to estimate each of the quantities in the table. In each case, the estimates matched the calculated values for the chosen L, $\widehat{\lambda}$, and $\overline{\lambda}$ to within a small error tolerance, indicating that our calculations are accurate.

4 Simulations

In this section, we compare the numerical performances of (8) with other proposed recursive methods. We also apply the method to the task of $O(N^2)$ SWC-RLS adaptive filtering and compare its numerical performance with an existing $O(N^2)$ method. In all cases, the algorithms have been implemented in the MATLAB signal manipulation environment. In addition, the MATLAB commands `randn()` and `filter()` have been used to generate zero-mean Gaussian random sequences with specific correlation statistics as test signals for the algorithms, and in the case of the sliding DFT methods, we have employed MATLAB's `fft()` routine to generate baseline quantities for numerical comparisons.

4.1 Sliding DFT

In this section, we explore the numerical performance of the sliding DFT in (2), the proposed sliding DFT in (8), and the LMS spectrum analyzer in [13], via simulations. In each case, we have computed the $L = 16$-point sliding DFT of a sequence of unit-variance Gaussian random variables using each method, and we then calculate the total squared numerical error between the resulting 16 signals and those obtained from a non-recursive FFT implementation at each time instant. For the proposed method, we have chosen $\overline{\lambda} = 0.999$. Figure 1 shows the total squared numerical errors for the three methods as a function of time, where we have only plotted the errors when $k \bmod L = L-1$ such that all of the methods would compute the exact DFT in an infinite-precision environment. As can be seen, the numerical errors of both the standard sliding DFT and LMS spectrum analyzer grow linearly over time, whereas that of the proposed technique does not exhibit such growth. This result indicates that the proposed technique mitigates numerical growth of errors, unlike the other recursive methods.

We now apply the proposed technique to transform-domain adaptive filtering, in which one wishes to estimate a component of a desired signal $d(k)$ using the output $\widehat{d}(k)$ of an adaptive FIR filter with N adjustable parameters in the vector $\mathbf{W}(k) = [w_0(k) \ \cdots \ w_{N-1}(k)]^T$. The output of

the filter is computed as

$$\widehat{d}(k) \;=\; \sum_{i=0}^{N} w_i(k)u_i(k) \;=\; \mathbf{W}^T(k)\mathbf{U}(k), \tag{31}$$

where $\mathbf{U}(k) = [u_0(k) \cdots u_{N-1}(k)]^T$ is a linearly-transformed version of the measured input signal vector $\mathbf{X}(k) = [x(k) \cdots x(k-N+1)]^T$. The goal of the adaptive filtering task is to adjust $\mathbf{W}(k)$ such that the mean-squared error

$$E\{|e(k)|^2\} \;=\; E\{|d(k) - \mathbf{W}^T(k)\mathbf{U}(k)|^2\} \tag{32}$$

is minimized over time. One algorithm for adjusting $\mathbf{W}(k)$ is the transform-domain least-mean-square (LMS) algorithm given by

$$\mathbf{W}(k+1) \;=\; \mathbf{W}(k) + \mathbf{M}(k)e(k)\mathbf{U}^*(k), \tag{33}$$

where $\mathbf{M}(k)$ is a diagonal matrix of positive step size parameters and $*$ denotes complex-conjugate. Typically, the (i,i)th value of $\mathbf{M}(k)$ is a scaled inverse of the time-averaged value of $|u_{i+1}(k)|^2$, as this choice causes the matrix $E\{\mathbf{M}(k)\mathbf{U}^*(k)\mathbf{U}^T(k)\}$ that governs the average adaptation behaviors of the coefficients to have a smaller eigenvalue spread and enables faster convergence of the coefficients to their optimum values [17, 18, 19].

Figure 2(a) and (b) shows the error signals $e(k)$ and $\overline{e}(k)$ of two different implementations of the adaptive filter in (33) as applied to an $N = L = 8$-coefficient linear prediction task in which $d(k) = x(k+1)$ and $x(k)$ is created by filtering a zero-mean Gaussian random sequence $s(k)$ as

$$x(k) \;=\; s(k) + 0.7s(k-1) + 1.2s(k-2) + 0.5s(k-3). \tag{34}$$

Shown for comparison are the errors from the systems in which the elements of $\mathbf{U}(k)$ are computed non-recursively and recursively via (8) with $\lambda = 0.999$, respectively. Each of the diagonal elements of $\mathbf{M}(k)$ for both algorithms are computed by filtering the squared value of each element of $\mathbf{U}(k)$ with a lowpass filter with transfer function $1/(1-0.99z^{-1})$ before inverting the resulting value. The two plots are nearly-indistinguishable from each other, and Figure 2(c) shows the difference signal $e(k) - \overline{e}(k)$, in which the magnitude of the difference is less than 0.1% over the entire adaptation period. Thus, the approximate sliding DFT method in (8) does not significantly alter the behavior of the overall system.

4.2 Recursive Least-Squares Adaptive Filtering

In sliding-window-covariance RLS adaptive filters, one minimizes the error criterion

$$\mathcal{J}(\mathbf{W}(k)) = \sum_{m=0}^{L-1} [d(k-m) - \mathbf{W}^T(k)\mathbf{X}(k-m)]^2, \tag{35}$$

where $d(k)$, $\mathbf{X}(k)$, and $\mathbf{W}(k)$ have been defined previously. By differentiating this cost function with respect to the filter coefficient vector $\mathbf{W}(k)$ and setting the result to the zero vector, we obtain the well-known solution

$$\mathbf{W}(k) = \mathbf{R}^{-1}(k)\mathbf{P}(k), \tag{36}$$

where

$$\mathbf{R}(k) = \sum_{m=0}^{L-1} \mathbf{X}(k-m)\mathbf{X}^T(k-m) \tag{37}$$

$$\mathbf{P}(k) = \sum_{m=0}^{L-1} d(k-m)\mathbf{X}(k-m). \tag{38}$$

Comparing $\mathbf{R}(k)$ and $\mathbf{P}(k)$ with $y(k)$ in (1), it is clear that we can recursively compute them as

$$\mathbf{R}(k) = \mathbf{R}(k-1) - \mathbf{X}(k-L)\mathbf{X}^T(k-L) + \mathbf{X}(k)\mathbf{X}^T(k) \tag{39}$$

$$\mathbf{P}(k) = \mathbf{P}(k-1) - d(k-L)\mathbf{X}(k-L) + d(k)\mathbf{X}(k), \tag{40}$$

in analogy with (2). Such a computational method is marginally-stable, however, and therefore, any implementation that explicitly or implicitly employs the update relationships in (39)–(40) experiences numerical error accumulation in a finite-precision environment.

Given our previous results, it is natural to consider a modified least-squares cost function of the form

$$\overline{\mathcal{J}}(\overline{\mathbf{W}}(k)) = \sum_{m=0}^{L-1} h_0(k,m)[d(k-m) - \overline{\mathbf{W}}^T(k)\mathbf{X}(k-m)]^2, \tag{41}$$

where $h_0(k,m)$ is as defined in (6). A similar derivation to the one that produces (36)–(40) can be performed for $\overline{\mathcal{J}}(\overline{\mathbf{W}}(k))$, and the solution obtained is identical to (36) except that the corresponding $\overline{\mathbf{R}}(k)$ and $\overline{\mathbf{P}}(k)$ are identical in form to $\overline{y}(k)$ in (5). Thus, in analogy with (8), we can compute $\overline{\mathbf{R}}(k)$ recursively as

$$\overline{\mathbf{R}}(k) = \begin{cases} \lambda[\overline{\mathbf{R}}(k) - \mathbf{X}(k-L)\mathbf{X}^T(k-L)] + \mathbf{X}(k)\mathbf{X}^T(k), & \text{if } k \bmod L = 0 \\ \overline{\mathbf{R}}(k) - \lambda\mathbf{X}(k-L)\mathbf{X}^T(k-L) + \mathbf{X}(k)\mathbf{X}^T(k), & \text{otherwise,} \end{cases} \tag{42}$$

and $\overline{\mathbf{P}}(k)$ has a similar update. The resulting recursive solution for $\overline{\mathbf{W}}(k)$ is numerically-stable, because the estimators for $\overline{\mathbf{R}}(k)$ and $\overline{\mathbf{P}}(k)$ are numerically-stable. Moreover, because $\overline{\mathcal{J}}(\overline{\mathbf{W}}(k))$ is identical to $\mathcal{J}(\mathbf{W}(k))$ for $k = nL - 1$, we have that

$$\overline{\mathbf{W}}(nL-1) \;=\; \mathbf{W}(nL-1), \tag{43}$$

in the absence of any numerical errors.

By applying the matrix inversion lemma [20] twice to the right-hand-side of (42), one can propagate the value of $\overline{\mathbf{R}}^{-1}(k)$ directly from $\overline{\mathbf{R}}^{-1}(k-1)$, $\mathbf{X}(k)$, and $\mathbf{X}(k-L)$. Then, by combining the updates with those for $\overline{\mathbf{P}}(k)$, one obtains a set of direct updates for $\overline{\mathbf{W}}(k)$. Table 2 lists the equations for this numerically-stable version of the $O(N^2)$ SWC-RLS algorithm, where $\overline{\mathbf{W}}(k) = \overline{\mathbf{W}}_L(k)$. Note that the complexity of this algorithm is identical to that of the original sliding-window-covariance RLS algorithm ($\lambda = 1$) when $k \bmod L \neq 0$, and thus for large values of L relative to the filter length N, the complexity of the proposed algorithm is similar to that of the original algorithm.

We have simulated the performance of both the original and proposed $O(N^2)$ SWC-RLS algorithms in the linear prediction task described previously, where $L = 16$ and $N = 8$. Figure 3 shows the total coefficient error powers $||\mathbf{W}(k) - \mathbf{W}_{opt}(k)||^2$ and $||\overline{\mathbf{W}}(k) - \mathbf{W}_{opt}(k)||^2$ for the original and proposed $O(N^2)$ SWC-RLS algorithms, respectively, where $\mathbf{W}_{opt}(k) = \mathbf{R}^{-1}(k)\mathbf{P}(k)$ is the non-recursive $O(N^3)$ least-squares solution at time k and $\overline{\lambda} = 0.995$ for the proposed algorithm. Here, we have plotted a smoothed version of the numerical errors at time instants where $k \bmod L = L - 1$. As can be seen, the original algorithm experiences numerical error accumulation over time, unlike the proposed method which remains numerically-stable.

5 Extensions

5.1 A Numerically-Stable Sliding DCT

While useful, the proposed technique for computing the sliding DFT in (8) employs complex-valued arithmetic, a feature that is undesirable in applications where $x(k)$ is real-valued. In such cases, a real-valued transform such as the discrete cosine transform (DCT) is more-useful. In addition, a recent analysis of the DFT and DCT in transform-domain adaptive filtering indicates that the DCT provides better orthogonalizing properties for certain input signals [19]. In this section, we provide a numerically-stable periodically-time-varying system that approximates the sliding DCT,

thus extending the method in (8) to linear systems that employ two marginally-stable complex-conjugate pole-zero cancellations.

A recursive method for computing the lth bin value of the sliding DCT for $1 \leq l \leq L-1$ is [17]

$$\begin{aligned} u_l(k) &= \gamma_l y(k) && (44) \\ y(k) &= 2\cos(l\pi/L)y(k-1) - y(k-2) + x(k) \\ &\quad - x(k-1) - (-1)^l[y(k-L) - y(k-L-1)], && (45) \end{aligned}$$

where $\gamma_l = \sqrt{2/L}\cos(l\pi/(2L))$. The computation of the zeroth sliding DCT bin value is the same as that for the sliding DFT in (2), except for a scale factor of $\gamma_0 = 1/\sqrt{L}$.

The corresponding periodically-time-varying system that implements the sliding DCT in a numerically-stable fashion replaces $y(k)$ in (44) by

$$\overline{y}(k) = \begin{cases} \overline{\lambda}[2\cos(l\pi/L)y(k-1) - y(k-2)] + x(k) \\ -x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \end{cases}, \text{ if } k\text{mod}L = 0$$
$$\begin{cases} 2\cos(l\pi/L)y(k-1) - \overline{\lambda}y(k-2) + x(k) \\ -x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \end{cases}, \text{ if } k\text{mod}L = 1 \quad (46)$$
$$\begin{cases} 2\cos(l\pi/L)y(k-1) - y(k-2) + x(k) \\ -x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \end{cases}, \text{ otherwise.}$$

In this case, a second-order recursive system is employed, and the stabilizing factor $\overline{\lambda}$ is applied to the recursive terms within the updates in different ways for two out of every L iterations of the system.

Figure 4 shows the total squared errors between the $L = 16$-point sliding DCT and the non-recursive DCT for both methods in (45) and (46), respectively, for $\overline{\lambda} = 0.999$. As in the previous cases, only the values at $k\text{mod}L = L-1$ are shown. We see that the growth in numerical errors for the original recursive implementation is mitigated by the proposed technique.

5.2 Arbitrary Duty Cycles

Because the proposed methods produce exactly the same results at time $k\text{mod}L = L-1$ as the original non-recursive methods in infinite-precision arithmetic, then we can easily extend the interval over which the value of $\overline{y}(k)$ matches that of $y(k)$ for each system in a simple fashion. For the DFT and DCT methods in (8) and (46), respectively, we replace the test $k\text{mod}L = 0$ with the test $k\text{mod}M = 0$, where M is any integer greater than L. Then, over the interval $L \leq k\text{mod}M < M$, we use the marginally-stable recursive updates (2) and (45) to compute the outputs of each system, respectively. In this way, the value of $\overline{y}(k)$ is mathematically-identical to the sliding DFT and DCT

over the periods defined by $L-1 \leq k \mathrm{mod} M < M$. A similar approach can be applied to the SWC-RLS algorithm in Table 2.

While potentially-useful, this extension reduces the effective number of stabilizing updates by the factor L/M over the original method, and thus one can expect that the levels of numerical error in the outputs will increase by a factor of approximately M/L. Thus, a smaller value of $\overline{\lambda}$ is necessary to obtain the same numerical performance as that of the original method, which causes a larger deviation between $\overline{y}(k)$ and $y(k)$ during the period $0 \leq k \mathrm{mod} M < L-1$.

6 Conclusions

In this paper, we have presented recursive systems that implement the sliding DFT and DCT in a numerically-stable fashion. These algorithms employ periodically-time-varying linear systems that mitigate the growth in numerical errors when implemented in finite-precision arithmetic while exactly computing the desired quantities over a portion of the input sequence. Moreover, we have applied this technique to the $O(N^2)$ sliding-window-covariance RLS adaptive filter, showing that the linear accumulation of numerical errors of the original system does not occur in the proposed system. These methods are expected to have wide use in a number of practical signal processing systems and in the implementation of adaptive filtering algorithms in particular.

While the techniques proposed in this paper are useful, a general method for deriving a periodically-time-varying system that exactly implements any particular marginally-stable time-invariant system has not been provided. Such a method would be useful, although the resulting systems may be quite complicated to implement in the most-general cases. In this regard, the techniques provided in this paper provide simple, effective solutions for several systems that are already in use in real-world situations.

References

[1] S. Kay, *Modern Spectral Estimation: Theory and Application* (Englewood Cliffs, NJ: Prentice-Hall, 1988).

[2] J.J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Mag.*, vol. 9, no. 1, pp. 14-37, Jan. 1992.

[3] S. Gay and S. Tavathia, "The fast affine projection algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Detroit, MI, vol. 5, pp. 3023-3026, May 1995.

[4] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Detroit, MI, vol. 2, pp. 945-948, May 1995.

[5] J. Benesty and P. Duhamel, "A fast exact least mean square adaptive algorithm," *IEEE. Trans. Signal Processing*, vol. 40, pp. 2904-2912, Dec. 1992.

[6] Q. Zhu, S.C. Douglas, and K.F. Smith, "A pipelined architecture for LMS adaptive filters without adaptation delay," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Munich, Germay, vol. 3, pp. 1933-1936, Apr. 1997.

[7] S.C. Douglas, Q. Zhu, and K.F. Smith, "A pipelined LMS adaptive FIR filter architecture without adaptation delay," accepted for publication in *IEEE Trans. Signal Processing*, to appear.

[8] S.C. Douglas, "The fast affine projection algorithm for active noise control," *Proc. 29th Asilomar Conf. Signals, Syst., Comput.,* Pacific Grove, CA, vol. 2, pp. 1245-1249, Oct. 1995.

[9] S.C. Douglas, "Fast exact filtered-X LMS and LMS algorithms for multichannel active noise control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Munich, Germany, vol. 3, pp. 1933-1936, Apr. 1997.

[10] S.C. Douglas, "An efficient implementation of the modified filtered-X LMS algorithm," *IEEE Signal Processing Lett.*, vol. 4, pp. 286-288, Oct. 1997.

[11] J.M. Cioffi and T. Kailath, "Windowed fast transversal filter adaptive algorithms with normalization," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, pp. 607-625, June 1985.

[12] D.T.M. Slock, "Underdetermined growing and sliding window covariance fast transversal filter RLS algorithms," *Proc. Signal Processing VI (EUSIPCO)*, Brussels, Belgium, vol. 2, pp. 1169-1172, Aug. 1992.

[13] B. Widrow, P. Baudrenghien, M. Vetterli, and P.F. Titchener, "Fundamental relations between the LMS algorithm and the DFT," *IEEE Trans. Circ. Syst.,* vol. 34, pp. 814-819, July 1987.

[14] F. Beaufays and B. Widrow, "On the advantages of the LMS spectrum analyzer over nonadaptive implementations of the sliding-DFT," *IEEE Trans. Circuits Systems II: Fund. Theory App.,* vol. 42, pp. 218-220, Apr. 1995.

[15] M.M. Covell, "An algorithm design environment for signal processing," Ph.D. thesis, Research Laboratory of Electronics, MIT, Cambridge, MA, 1989.

[16] B. Farhang-Boroujeny, "$O(N)$-complexity transform domain adaptive filters," *IEEE Trans. Circ. Syst. II: Anal. Dig. Signal Processing,* vol. 42, pp. 478-480, July 1995.

[17] S.S. Narayan and A.M. Peterson, "Frequency domain least-mean-square algorithm," *Proc. IEEE,* vol. 69, pp. 124-126, Jan. 1981.

[18] D.F. Marshall, W.K. Jenkins, and J.J. Murphy, "The use of orthogonal transforms for improving the performance of adaptive filters," *IEEE Trans. Circ. Syst.,* vol. 36, pp. 474-484, Apr. 1989.

[19] F. Beaufays, "Transform-domain adaptive filters: an analytical approach," *IEEE Trans. Signal Processing,* vol. 43, pp. 422-431, Feb. 1995.

[20] T. Kailath, *Linear Systems Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1980).

List of Tables

List of Figures

List of Footnotes

Manuscript received_____.

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

†Please address correspondence to: Scott C. Douglas, Department of Electrical Engineering, University of Utah, 50 South Central Campus Drive, Room 3280, Salt Lake City, UT 84112. (801) 581-4445. FAX: (801) 581-5281. Electronic mail address: douglas@ee.utah.edu. World Wide Web URL: http://www.elen.utah.edu/~douglas.

Table 1: Comparison of numerical performance, exponentially-windowed and proposed schemes, $l = 0$.

Window	*Parameter*		*Bias*		*Error Variance*		*Precision*
L	$1-\widehat{\lambda}$	$1-\overline{\lambda}$	$\widehat{\mu}/(L\mu_x)$	$\overline{\mu}/(L\mu_x)$	$\widehat{\sigma}^2/(L\sigma_x^2)$	$\overline{\sigma}^2/(L\sigma_x^2)$	$\frac{\sigma_\epsilon^2(\infty)}{(L\sigma_x^2)}$
10	1.81×10^{-3}	0.01	8.11×10^{-3}	4.50×10^{-3}	9.24×10^{-5}	4.50×10^{-5}	55.3
100	1.97×10^{-4}	0.01	9.69×10^{-3}	4.95×10^{-3}	1.26×10^{-4}	4.95×10^{-5}	50.8
$\to\infty$	$\to 0$	0.01	9.89×10^{-3}	5×10^{-3}	1.30×10^{-4}	5×10^{-5}	50.3
10	1.82×10^{-4}	0.001	8.18×10^{-4}	4.50×10^{-4}	9.40×10^{-7}	4.50×10^{-7}	550
100	1.98×10^{-5}	0.001	9.79×10^{-4}	4.95×10^{-4}	1.28×10^{-6}	4.95×10^{-7}	505
$\to\infty$	$\to 0$	0.001	9.99×10^{-4}	5×10^{-4}	1.33×10^{-6}	5×10^{-7}	500
10	1.82×10^{-5}	1×10^{-4}	8.18×10^{-5}	4.50×10^{-5}	9.42×10^{-9}	4.50×10^{-9}	5500
100	1.98×10^{-6}	1×10^{-4}	9.80×10^{-5}	4.95×10^{-5}	1.29×10^{-8}	4.95×10^{-9}	5050
$\to\infty$	$\to 0$	1×10^{-4}	1.00×10^{-5}	5×10^{-5}	1.33×10^{-8}	5×10^{-9}	5000

Table 2: The $O(N^2)$ stabilized SWC-FTF algorithm.

#	*Equation*
	if $k \bmod L = 0$
1(a)	$\mathbf{C}(k) = \dfrac{\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}{\overline{\lambda}+\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}$
2(a)	$\mathbf{R}_{L+1}^{-1}(k) = \overline{\lambda}^{-1}\left(\mathbf{R}_L^{-1}(k-1)-\mathbf{C}(k)\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\right)$
	else
1(b)	$\mathbf{C}(k) = \dfrac{\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}{1+\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}$
2(b)	$\mathbf{R}_{L+1}^{-1}(k) = \mathbf{R}_L^{-1}(k-1)-\mathbf{C}(k)\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)$
	end
3	$\epsilon(k) = d(k)-\mathbf{X}^T(k)\overline{\mathbf{W}}_L(k-1)$
4	$\overline{\mathbf{W}}_{L+1}(k) = \overline{\mathbf{W}}_L(k-1)+\epsilon(k)\mathbf{C}(k)$
5	$\mathbf{D}(k) = \mathbf{R}_{L+1}^{-1}(k)\mathbf{X}(k-L)$
6	$\gamma(k) = \left(-\overline{\lambda}^{-1}+\mathbf{X}^T(k-L)\mathbf{D}(k)\right)^{-1}$
7	$\mathbf{R}_L^{-1}(k) = \mathbf{R}_{L+1}^{-1}(k)-\mathbf{D}(k)\mathbf{D}^T(k)\gamma(k)$
8	$\nu(k) = d(k-L)-\mathbf{X}^T(k-L)\overline{\mathbf{W}}_{L+1}(k)$
9	$\overline{\mathbf{W}}_L(k) = \overline{\mathbf{W}}_{L+1}(k)+\gamma(k)\nu(k)\mathbf{D}(k)$

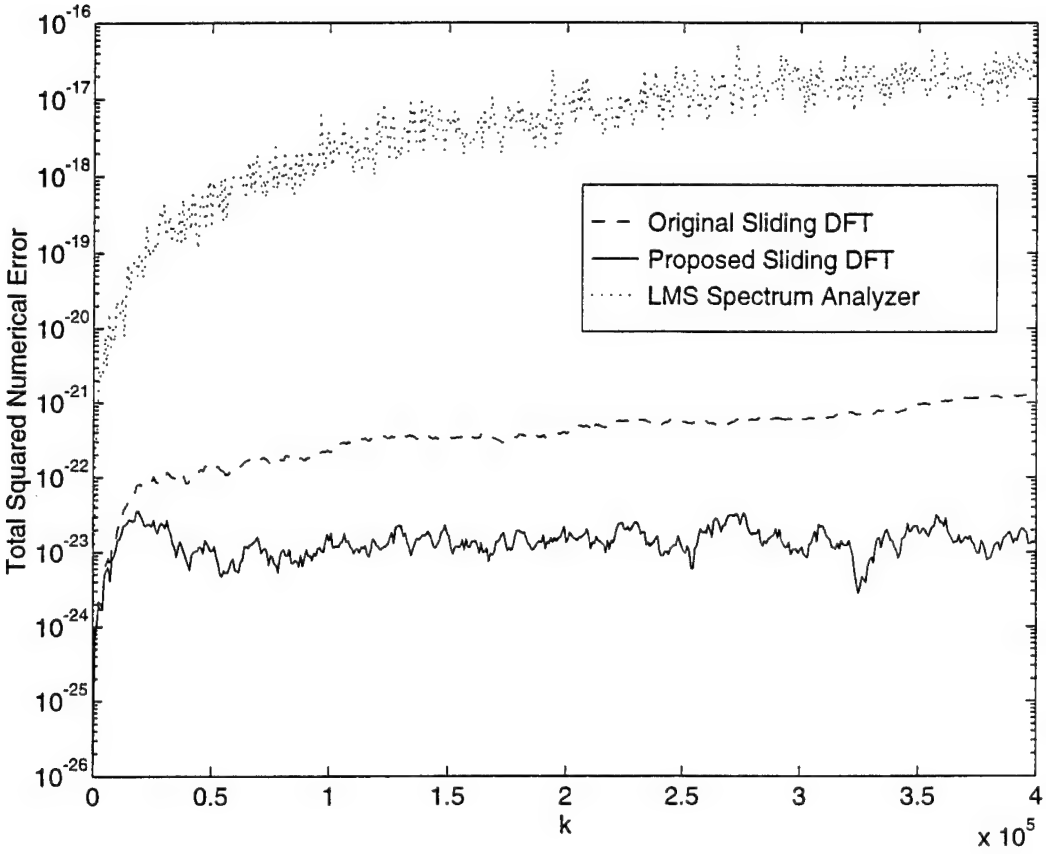

Figure 1: Total squared numerical errors: original sliding DFT, proposed sliding DFT, and the LMS spectrum analyzer, uncorrelated Gaussian input signals, $L = 16$, $\overline{\lambda} = 0.999$.

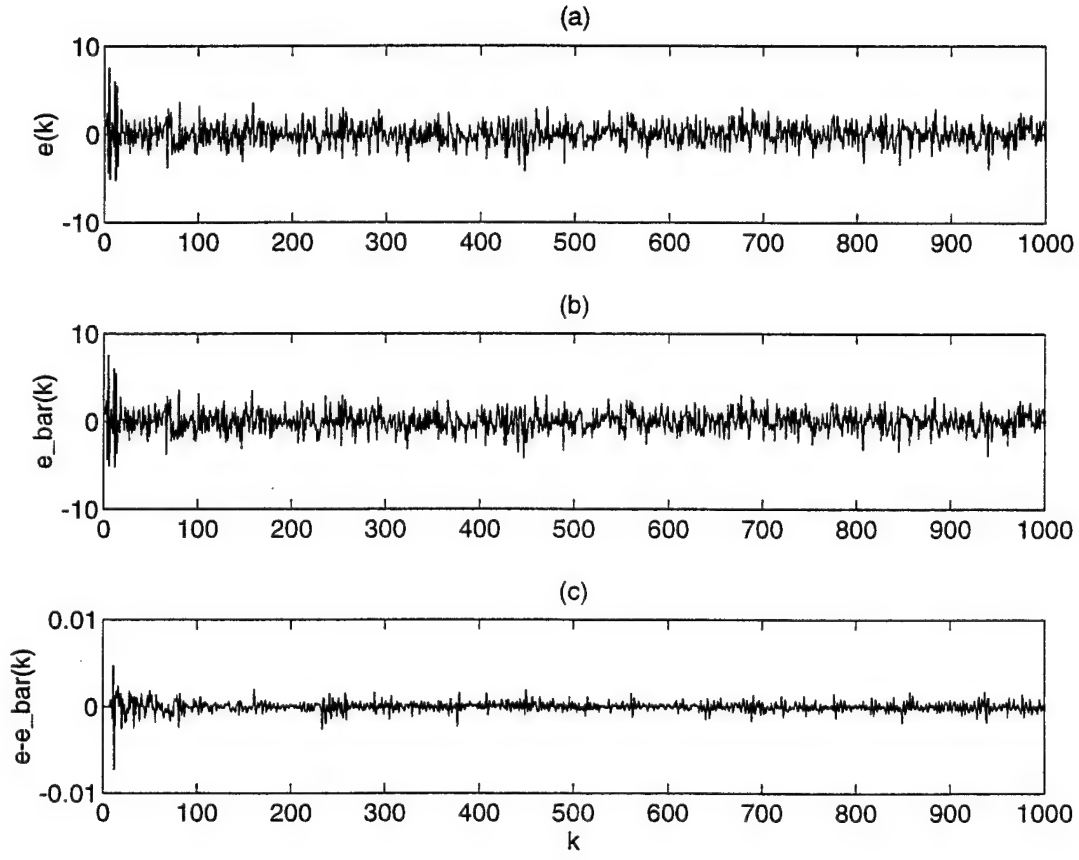

Figure 2: The behaviors of the non-recursive and proposed sliding DFTs in the transform-domain adaptive filtering example.

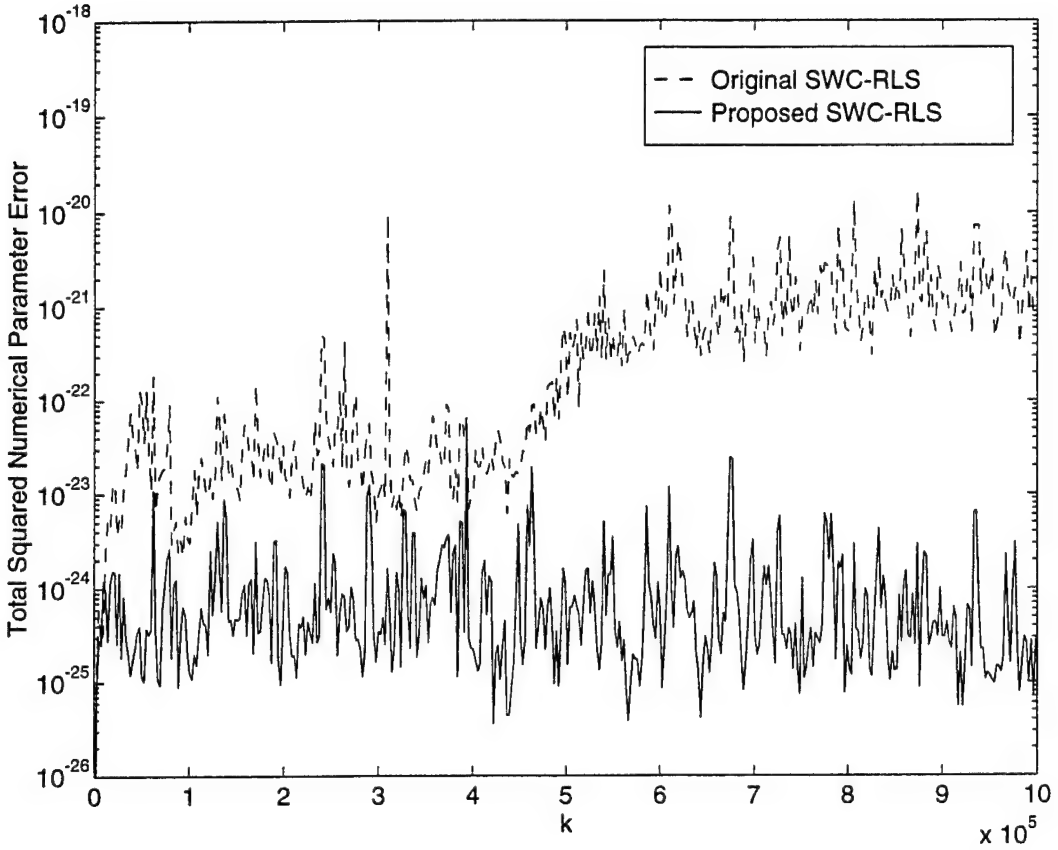

Figure 3: Total squared numerical errors in estimated coefficients: standard and proposed $O(N^2)$ SWC-RLS adaptive filters, $L = 16$, $N = 8$, $\overline{\lambda} = 0.995$.

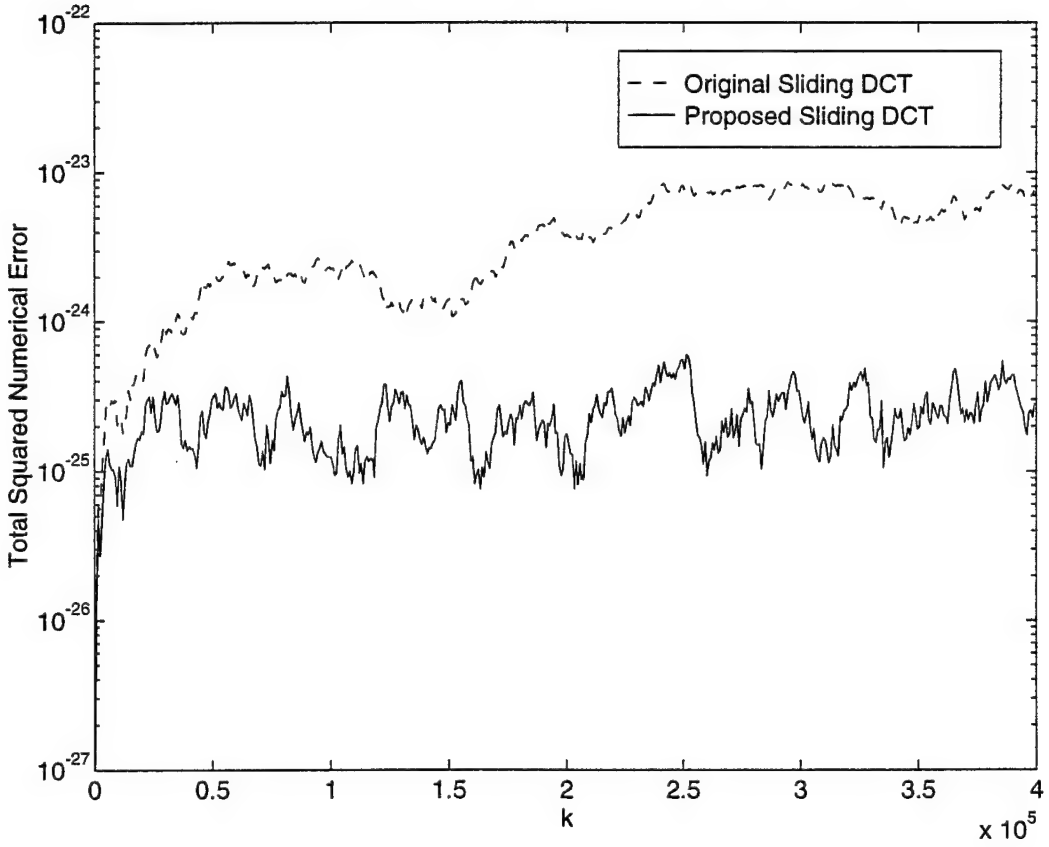

Figure 4: Total squared numerical errors: original and proposed sliding DCTs, uncorrelated Gaussian input signals, $L = 16$, $\overline{\lambda} = 0.999$.

An Efficient Implementation of the Modified Filtered-X LMS Algorithm

Scott C. Douglas, *Member, IEEE*

***Abstract*—The filtered-X LMS algorithm used in active noise control applications suffers from slow convergence due in part to the output delay caused by the secondary path of the system. In this letter, we provide a computationally efficient method for removing this output delay within the error signal used to update the controller coefficients, such that the standard LMS algorithm is effectively employed.**

***Index Terms*— Acoustic signal processing, adaptive control, adaptive filters, adaptive signal processing, least mean square methods.**

I. Summary

THE filtered-X least-mean-square (LMS) algorithm is a useful method for adapting the coefficients of a finite impulse response (FIR) filter for active noise control applications [1]. The coefficient updates for this algorithm are given by

$$\mathbf{W}_L(n+1) = \mathbf{W}_L(n) - \mu(n)\epsilon(n)\mathbf{F}_L(n) \tag{1}$$

where $\mathbf{W}_L(n) = [w_0(n) \cdots w_{L-1}(n)]^T$ is the coefficient vector of the adaptive filter at time n, $\mathbf{F}_L(n) = [f(n) \cdots f(n-L+1)]^T$ contains the L most-recent samples of the filtered input signal, $\epsilon(n)$ is the error sensor signal at time n, and $\mu(n)$ is the step size at time n. In this case, $f(n)$ is computed as

$$f(n) = \mathbf{H}_M^T\mathbf{X}_M(n-1) \tag{2}$$

where $\mathbf{H}_M[h_1 \cdots h_M]^T$ contains the secondary path impulse response and $\mathbf{X}_M(n) = [x(n) \cdots x(n-M+1)]^T$ contains the M most recent samples of the input signal. The output of the controller is computed as

$$y(n) = \mathbf{X}_L^T(n)\mathbf{W}_L(n) \tag{3}$$

where $\mathbf{X}_L(n) = [x(n) \cdots x(n-L+1)]^T$ contains the L most-recent samples of the input signal. The error sensor signal $\epsilon(n)$ is the sum of the primary signal $d(n)$ to be cancelled and a filtered version of the controller output, given by

$$\epsilon(n) = d(n) + \mathbf{H}_M^T\mathbf{Y}_M(n-1) \tag{4}$$

Manuscript received June 3, 1996. This material is based upon work supported in part by the U.S. Army Research Office under Contract DAAH04-96-1-0085, and in part by the NSF under Grant MIP-9501680. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. G. W. Elko.

The author is with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA (e-mail: douglas@ee.utah.edu).

Publisher Item Identifier S 1070-9908(97)07512-3.

where $\mathbf{Y}_M(n) = [y(n) \cdots y(n-M+1)]^T$ and where we have assumed that the FIR filter $\mathbf{H}_M$ accurately represents the secondary path transfer characteristics. Since (4) is not explicitly calculated within the controller, the filtered-X LMS adaptive filter requires $2L + M + 1$ multiply/accumulates (MAC's) per iteration, and it uses $3L + M + 2$ memory locations.

By substituting (3) into (4), we see that the error sensor signal is effectively computed using past coefficient vectors $\{\mathbf{W}_L(n-1), \cdots, \mathbf{W}_L(n-M)\}$. The resulting delay within the coefficient updates leads to slower convergence, a restricted step size range for stability, and overall reduced performance [2]. A method is presented in [3]–[5] to effectively remove this delay within the coefficient updates, such that the behavior of the system is similar to that of a standard LMS adaptive filter with input signal $f(n)$ and desired response signal $d(n)$. The coefficient vector update for this modified filtered-X LMS algorithm is

$$\mathbf{W}_L(n+1) = \mathbf{W}_L(n) - \mu(n)e(n)\mathbf{F}_L(n) \tag{5}$$

$$e(n) = \epsilon(n) - \mathbf{H}_M^T\mathbf{Y}_M(n-1) + \mathbf{F}_L^T(n)\mathbf{W}_L(n). \tag{6}$$

In this algorithm, it is assumed that $\mathbf{H}_M$ in (6) accurately represents the secondary path impulse response so that the computed value of $\epsilon(n) - \mathbf{H}_M^T\mathbf{Y}_M(n-1)$ accurately represents the primary disturbance $d(n)$ to be cancelled. This algorithm requires $3L + 2M + 1$ MAC's per iteration and $3L + 2M + 3$ memory locations to implement as described.

For long filter lengths L relative to the secondary path impulse response length M, the additional complexity of the modified filtered-X LMS algorithm over that of the filtered-X LMS algorithm is prohibitive in some cases. Approximate implementations of the above algorithm have been developed that use approximately $2L + 2M$ MAC's per iteration [6]; however, the performance of these algorithms only approaches that of the above algorithm in certain cases. An alternate, exact implementation of the algorithm in (5) and (6) is described in [7]; however, this implementation requires about $2L + 0.5M^2 + 2.5M$ MAC's per iteration, which for large M is computationally prohibitive.

In this paper, we derive a novel exact implementation of the algorithm in (5) and (6). Our equivalent implementation requires only $2L + 5M + 1$ MAC's per iteration and $3L + 4M + 3$ memory locations to implement. Thus, if $L > 3M$, our implementation is more computationally efficient than the original implementation, and for $M \geq 5$, our implementation is more computationally efficient than that presented in [7].

TABLE I
FAST MODIFIED FILTERED-X LMS ALGORITHM

#	*Equation*	*# MAC's*
1	$e^{(\mu)}(n) = \mu(n)\{\epsilon(n) + \mathbf{H}_M^T \mathbf{U}_M(n-1)\}$	$M+1$
2	$f(n) = \mathbf{H}_M^T \mathbf{X}_M(n-1)$	M
3	$y(n) = \mathbf{X}_L^T(n)\mathbf{W}_L(n)$	L
4	$\mathbf{F}_L(n) = \begin{bmatrix} f(n) \\ \mathbf{F}_{L-1}(n-1) \end{bmatrix}$	0
5	$\mathbf{R}_M(n) = \mathbf{R}_M(n-1) + \mathbf{X}_M(n)f(n) - \mathbf{X}_M(n-L)f(n-L)$	$2M$
6	$\mathbf{U}_M(n) = \begin{bmatrix} 0 \\ \mathbf{U}_{M-1}(n-1) \end{bmatrix} - \mathbf{R}_M(n)e^{(\mu)}(n)$	M
7	$\mathbf{W}_L(n+1) = \mathbf{W}_L(n) - e^{(\mu)}(n)\mathbf{F}_L(n)$	L
Total: $2L+5M+1$		

II. ALGORITHM DERIVATION

The algorithm we derive is a nontrivial modification of a method presented in [8] to remove the adaptation delay within the delayed LMS algorithm. For this derivation, we note that the additional complexity of the modified filtered-X LMS algorithm over that of the filtered-X LMS algorithm stems from the error recalculation in (6). By defining the matrix

$$X(n) = [\mathbf{X}_L(n) \cdots \mathbf{X}_L(n-M+1)] \tag{7}$$

$$= [\mathbf{X}_M(n) \cdots \mathbf{X}_M(n-L+1)]^T \tag{8}$$

we can express $e(n)$ as

$$e(n) = \epsilon(n) - \mathbf{H}_M^T \mathbf{Y}_M(n-1) + \mathbf{H}_M^T X^T(n-1)\mathbf{W}_L(n) \tag{9}$$

$$= \epsilon(n) + \mathbf{H}_M^T \mathbf{U}_M(n-1) \tag{10}$$

with the M-dimensional vector $\mathbf{U}_M(n)$ defined as

$$\mathbf{U}_M(n) = X^T(n)\mathbf{W}_L(n+1) - \mathbf{Y}_M(n). \tag{11}$$

To efficiently compute the vector $\mathbf{U}_M(n)$, we see that

$$\mathbf{U}_M(n) = \begin{bmatrix} \mathbf{X}_L^T(n)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n)\} \\ \mathbf{X}_L^T(n-1)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n-1)\} \\ \vdots \\ \mathbf{X}_L^T(n-M+1)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n-M+1)\} \end{bmatrix}. \tag{12}$$

We can write $\mathbf{U}_M(n)$ as

$$\mathbf{U}_M(n) = \begin{bmatrix} 0 \\ \mathbf{X}_L^T(n-1)\{\mathbf{W}_L(n) - \mathbf{W}_L(n-1)\} \\ \vdots \\ \mathbf{X}_L^T(n-M+1)\{\mathbf{W}_L(n) - \mathbf{W}_L(n-M+1)\} \end{bmatrix} + \begin{bmatrix} \mathbf{X}_L^T(n)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n)\} \\ \mathbf{X}_L^T(n-1)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n)\} \\ \vdots \\ \mathbf{X}_L^T(n-M+1)\{\mathbf{W}_L(n+1) - \mathbf{W}_L(n)\} \end{bmatrix} \tag{13}$$

$$= \begin{bmatrix} 0 \\ \mathbf{U}_{M-1}(n-1) \end{bmatrix} - X^T(n)\mathbf{F}_L(n)e^{(\mu)}(n) \tag{14}$$

$$e^{(\mu)}(n) = \mu(n)e(n) \tag{15}$$

where $\mathbf{U}_{M-1}(n)$ contains the first $M-1$ elements of $\mathbf{U}_M(n)$ and we have substituted the update relationship for $\mathbf{W}_L(n)$ in (5) into the second term of (13). Define the M-dimensional vector $\mathbf{R}_M(n)$ as

$$\mathbf{R}_M(n) = X^T(n)\mathbf{F}_L(n). \tag{16}$$

Then, it can be shown that $\mathbf{R}_M(n)$ can be updated as

$$\mathbf{R}_M(n) = \mathbf{R}_M(n-1) + \mathbf{X}_M(n)f(n) - \mathbf{X}_M(n-L)f(n-L). \tag{17}$$

Combining (2), (3), (5), (10), (14), (15), and (17) gives the fast implementation of the modified filtered-X LMS algorithm. Table I lists the complete algorithm and the number of multiplies required for each step, where $\mathbf{F}_{L-1}(n)$ contains the $L-1$ most recent samples of the filtered input signal $f(n)$. The overall complexity of this new implementation is less than that of the original implementation if $L > 3M$. Such a situation is reasonable in broadband active noise control applications when the secondary source is located near the error sensor and both the secondary source and error sensor are far away from the input sensor.

III. CONCLUSION

In this letter, we have shown how the modified filtered-X LMS algorithm can be implemented efficiently, so that standard delayless LMS adaptation can be employed in active noise control applications. The modification described here can also be applied to other useful algorithms for active noise control, such as those employing sgn $(\cdot)$-type error and data nonlinearities [9].

REFERENCES

[1] S. M. Kuo and D. R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*. New York: Wiley, 1996.
[2] E. Bjarnason, "Analysis of the filtered-X LMS algorithm," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 504–514, Nov. 1995.

[3] ——, "Active noise cancellation using a modified form of the filtered-X LMS algorithm," in *Proc. Sixth Europ. Signal Processing Conf.* Amsterdam: Elsevier, 1992, vol. 2, pp. 1053–1056.

[4] M. Rupp and R. Frenzel, "The behavior of LMS and NLMS algorithms with delayed coefficient update in the presence of spherically invariant processes," *IEEE Trans. Signal Processing*, vol. 42, pp. 668–672, Mar. 1994.

[5] I. Kim, H. Na, K. Kim, and Y. Park, "Constraint filtered-X and filtered-U algorithms for the active control of noise in a duct," *J. Acoust. Soc. Amer.*, vol. 95, pp. 3397–3389, June 1994.

[6] M. Rupp and A. H. Sayed, "Modified FxLMS algorithms with improved convergence performance," in *Conf. Rec. 29th Asilomar Conf. Signals. Systems, Computers.* Los Alamitos, CA: IEEE Comput. Soc. Press, 1995, vol. 2, pp. 1255–1259.

[7] M. Rupp, "Saving complexity of modified filtered-X LMS and delayed update LMS algorithms," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 45–48, Jan. 1997.

[8] R. D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," *IEEE Signal Processing Lett.*, vol. 2, p. 223, Dec. 1995.

[9] D. L. Duttweiler, "Adaptive filter performance with nonlinearities in the correlation multiplier," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 578–586, Aug. 1982.

NOISE-CON 97

The Pennsylvania State University
University Park, Pennsylvania
1997 June 15-17

FAST BLOCK ADAPTIVE ALGORITHMS FOR FEEDFORWARD ACTIVE NOISE CONTROL

David S. Nelson, Scott C. Douglas, and Marc Bodson

Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112 USA

I. INTRODUCTION

In single-channel feedforward active noise control, an input signal $x(n)$ is measured and processed by a time-varying finite-impulse-response (FIR) filter as

$$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l) = \mathbf{X}^T(n)\mathbf{W}(n), \quad (1)$$

where the vector $\mathbf{X}(n) = [x(n) \cdots x(n-L+1)]^T$ contains the L most-recent samples of the input signal and $\mathbf{W}(n) = [w_0(n) \cdots w_{L-1}(n)]^T$ is the coefficient vector of the controller at time n. The output of the controller propagates to the desired quiet region, where it combines with the primary signal to be cancelled, denoted by $d(n)$. An error sensor measures the linear combination of these two signals as modeled by the equation

$$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T\underline{\mathbf{Y}}(n-1), \quad (2)$$

where $\epsilon(n)$ is the measured error signal, $\underline{\mathbf{Y}}(n-1) = [y(n-1) \cdots y(n-M)]^T$, and where the FIR filter $\underline{\mathbf{H}} = [h_1 \cdots h_M]^T$ is assumed to accurately represent the transfer characteristics of the output-actuator-to-error-sensor secondary path. The task is to adjust or adapt $\mathbf{W}(n)$ so that the magnitude of $\epsilon(n)$ is reduced over time.

Various algorithms for adjusting $\mathbf{W}(n)$ have been proposed [1, 2, 3, 4]. For the most part, they involve multiply and add operations that are similar to the FIR filter output calculation, and they typically employ an estimate of the secondary path impulse response $\underline{\mathbf{H}}$. To obtain higher cancellation performance in broadband noise control applications, it is desirable to increase the lengths of $\mathbf{W}(n)$ and $\underline{\mathbf{H}}$ so that the controller can more accurately represent the physical paths within the system. This change increases the number of computations to be performed by the controller at each time instant. For fixed computational resources and sampling rates, however, such increases can exceed the limited time available to perform the necessary calculations. Thus, if longer FIR filters are desired, then either a change in the algorithm is required or a new implementation of the algorithm is needed.

In this paper, we apply and extend a recently-developed method for computing convolutions via block updates [6, 7] to five different algorithms for single-channel feedforward active noise control. By choosing a block length of two, we guarantee that the input-output characteristics

of the original algorithms are unchanged in our new implementations, unlike other proposed block update methods that introduce significant delays into the coefficient updates [5]. In many cases, the new implementations use approximately 25% fewer multiplications as compared to their original implementations, and they require fewer additions as well. Thus, our implementations should allow for longer controller filters for fixed computational resources and sampling rates while still guaranteeing equivalent controller behavior.

The organization of the paper is as follows. In Section II, we describe the filtered–X LMS algorithm [1]. We then describe the fast convolution method employing block-of-two operations and apply the method to the filtered–X LMS algorithm. In Sections III, IV, V, and VI, we apply similar methods to the active noise control LMS (ANC-LMS) algorithm [2], the fast ANC-LMS algorithm [3], the periodic filtered–X LMS algorithm [4], and the sequential filtered–X LMS algorithm [4]. Tables of the new implementations are provided in each case, and an overall comparison of the algorithms' complexities is given in Section VII.

II. THE BLOCK-UPDATE FILTERED–X LMS ALGORITHM

In the filtered–X LMS algorithm for active noise control, the coefficient updates are given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu(n)\epsilon(n)\mathbf{F}(n), \tag{3}$$

where $\mathbf{W}(n)$ and $\epsilon(n)$ are as defined before, $\mathbf{F}(n) = [f(n) \cdots f(n-L+1)]^T$ contains the L most-recent samples of the filtered input signal, and $\mu(n)$ is the step size at time n. In this system, $f(n)$ is computed as

$$f(n) = \underline{\mathbf{H}}^T \underline{\mathbf{X}}(n-1), \tag{4}$$

where $\underline{\mathbf{X}}(n-1) = [x(n-1) \cdots x(n-M)]^T$ contains M samples of the input signal. The output of the controller is computed as in (1). Considering (1), (3), and (4), we find that this algorithm requires $2L+M+1$ multiplications and $2L+M-2$ additions per time step, and it employs $2L+M+\max(L, M+1)+3$ memory locations within the controller.

We now describe a method for simplifying the implementation of a fixed-coefficient FIR filter that calculates terms that can be reused at successive time instants to calculate the output signal [6]. Two successive outputs of a FIR filter with fixed coefficient vector $\mathbf{W}$ can be computed as

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \mathbf{W} = \begin{bmatrix} \tilde{\mathbf{X}}^T(n-1) & \tilde{\mathbf{X}}^T(n-2) \\ \tilde{\mathbf{X}}^T(n) & \tilde{\mathbf{X}}^T(n-1) \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{W}}_0 \\ \widetilde{\mathbf{W}}_1 \end{bmatrix}, \tag{5}$$

where

$$\tilde{\mathbf{X}}(n) = [x(n)\, x(n-2) \cdots x(n-L+2)]^T \tag{6}$$

$$\widetilde{\mathbf{W}}_i = [w_i\, w_{2+i} \cdots w_{L-2+i}]^T. \tag{7}$$

Expanding the last term on the right-hand-side of (5), we get

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0 + \widetilde{\mathbf{W}}_1) + (\tilde{\mathbf{X}}^T(n-2) - \tilde{\mathbf{X}}^T(n-1))\widetilde{\mathbf{W}}_1 \\ \tilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0 + \widetilde{\mathbf{W}}_1) - (\tilde{\mathbf{X}}^T(n-1) - \tilde{\mathbf{X}}^T(n))\widetilde{\mathbf{W}}_0 \end{bmatrix} \tag{8}$$

$$= \begin{bmatrix} x_w(n-1) + \tilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1 \\ x_w(n-1) - \tilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0 \end{bmatrix}, \tag{9}$$

where we have defined

$$x_w(n) = \tilde{\mathbf{X}}^T(n)(\widetilde{\mathbf{W}}_0 + \widetilde{\mathbf{W}}_1) \tag{10}$$

$$\tilde{\mathbf{X}}_\Delta(n) = \tilde{\mathbf{X}}(n-1) - \tilde{\mathbf{X}}(n). \tag{11}$$

Computing the right-hand side of (9) requires $1.5L$ multiplications, since the multiplications needed to compute $x_w(n)$ are common to both time steps and are performed only once. Therefore, this implementation of the fixed-coefficient FIR filter uses $0.75L$ multiplies per time step on average, a savings of $0.25L$ multiplies over the standard FIR filter implementation. Moreover, storing the difference terms $x(n-i)-x(n-i-1)$ in $\tilde{\mathbf{X}}_\Delta(n)$ reduces the number of additions needed to compute the filter output at the expense of additional memory.

When $\mathbf{W}(n)$ is time-varying, the coefficient updates must be included within the above calculations. In [7], a method for calculating the outputs and updates for the LMS adaptive filter in a block fashion is described. We now extend this method to the filtered-X LMS algorithm. First, we rewrite equations (1) and (3) at time $n-1$, which gives

$$y(n-1) = \mathbf{X}^T(n-1)\mathbf{W}(n-1) \tag{12}$$

$$\mathbf{W}(n) = \mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)\mathbf{F}(n-1)\,. \tag{13}$$

Substituting equation (13) into equation (1), we obtain

$$y(n) = \mathbf{X}^T(n)\left(\mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)\mathbf{F}(n-1)\right) \tag{14}$$

$$= \mathbf{X}^T(n)\mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)r_{xf}(n)\,, \tag{15}$$

where we have defined

$$r_{xf}(n) = \mathbf{X}^T(n)\mathbf{F}(n-1)\,. \tag{16}$$

Considering two successive output samples, we can write

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \mathbf{W}(n-1) - \begin{bmatrix} 0 \\ \mu(n-1)\epsilon(n-1)r_{xf}(n) \end{bmatrix}. \tag{17}$$

The first term on the right-hand-side of (17) has the same form as that in (5). Therefore, we can use (9) to write this relationship as

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} x_w(n-1) + \tilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1) \\ x_w(n-1) - \tilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) \end{bmatrix} - \begin{bmatrix} 0 \\ \mu(n-1)\epsilon(n-1)r_{xf}(n) \end{bmatrix}, \tag{18}$$

with

$$\widetilde{\mathbf{W}}_i(n) = [w_i(n)\; w_{2+i}(n)\; \cdots\; w_{L-2+i}(n)]^T \tag{19}$$

$$x_w(n) = \tilde{\mathbf{X}}^T(n)(\widetilde{\mathbf{W}}_0(n) + \widetilde{\mathbf{W}}_1(n))\,. \tag{20}$$

To compute $r_{xf}(n)$ in an efficient manner, we note that

$$r_{xf}(n) = \sum_{i=0}^{L} x(n-i)f(n-i-1) \tag{21}$$

$$= r_{xf}(n-1) + x(n)f(n-1) - x(n-L)f(n-L-1) \tag{22}$$

$$= r_{xf}(n-2) + \Delta_{xf}(n) - \Delta_{xf}(n-L)\,, \tag{23}$$

where

$$\Delta_{xf}(n) = x(n)f(n-1) + x(n-1)f(n-2)\,. \tag{24}$$

The procedure for computing $[f(n-1)\ f(n)]^T$ is similar to the original fast convolution method in (9), as the value of $\underline{\mathbf{H}}$ is fixed. The resulting equations are

$$\begin{bmatrix} f(n-1) \\ f(n) \end{bmatrix} = \begin{bmatrix} x_h(n-1) + \underline{\tilde{\mathbf{X}}}_\Delta^T(n-1)\underline{\tilde{\mathbf{H}}}_1 \\ x_h(n-1) - \underline{\tilde{\mathbf{X}}}_\Delta^T(n)\underline{\tilde{\mathbf{H}}}_0 \end{bmatrix}, \tag{25}$$

where we define

$$\begin{aligned} \underline{\tilde{\mathbf{X}}}_\Delta(n) &= \underline{\tilde{\mathbf{X}}}(n-1) - \underline{\tilde{\mathbf{X}}}(n) && (26)\\ \underline{\tilde{\mathbf{H}}}_i &= [h_{1+i}\ h_{3+i}\ \cdots\ h_{M-1+i}]^T && (27)\\ \underline{\tilde{\mathbf{X}}}(n) &= [x(n-1)\ x(n-3)\ \cdots\ x(n-M+1)]^T && (28)\\ x_h(n) &= \underline{\tilde{\mathbf{X}}}^T(n)\underline{\tilde{\mathbf{H}}}_\Delta && (29)\\ \underline{\tilde{\mathbf{H}}}_\Delta &= \underline{\tilde{\mathbf{H}}}_0 + \underline{\tilde{\mathbf{H}}}_1 . && (30) \end{aligned}$$

A similar method could be used to compute $[\epsilon(n-1)\ \epsilon(n)]^T$ for simulation purposes: in practice, however, $\epsilon(n)$ is measured by the error sensor and need not be computed.

To update the filter coefficients, we consider two successive iterations of (3) to obtain

$$\mathbf{W}(n+1) = (\mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)\mathbf{F}(n-1)) - \mu(n)\epsilon(n)\mathbf{F}(n). \tag{31}$$

or, equivalently,

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{F}}(n-1)\mu(n-1)\epsilon(n-1) + \tilde{\mathbf{F}}(n)\mu(n)\epsilon(n) \\ \tilde{\mathbf{F}}(n-2)\mu(n-1)\epsilon(n-1) + \tilde{\mathbf{F}}(n-1)\mu(n)\epsilon(n) \end{bmatrix} \tag{32}$$

$$= \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{F}}_\epsilon(n-1) - \tilde{\mathbf{F}}_\Delta(n)\mu(n)\epsilon(n) \\ \tilde{\mathbf{F}}_\epsilon(n-1) + \tilde{\mathbf{F}}_\Delta(n-1)\mu(n-1)\epsilon(n-1) \end{bmatrix}, \tag{33}$$

where we have defined

$$\begin{aligned} \tilde{\mathbf{F}}_\Delta(n) &= \tilde{\mathbf{F}}(n-1) - \tilde{\mathbf{F}}(n) && (34)\\ \tilde{\mathbf{F}}(n) &= [f(n)\ f(n-2)\ \cdots\ f(n-L+2)]^T && (35)\\ \tilde{\mathbf{F}}_\epsilon(n) &= \tilde{\mathbf{F}}(n)(\mu(n)\epsilon(n) + \mu(n+1)\epsilon(n+1)) . && (36) \end{aligned}$$

Table 1 list the equations for the block-of-two implementation of the filtered–X LMS algorithm along with the number of multiplications and additions needed for each step. In the table, a horizontal line separates the operations necessary at each time step; however, it is possible to shift some of the calculations from the second half of the block to the first half to balance the computational load of the algorithm at each sample time. This algorithm requires $3L + 1.5M + 7$ multiplications, $4L + 1.5M + 8$ additions, and $2.5L + 1.5M + 1.5\max(L, M+1) + 14$ memory locations per two sample times. Thus, the maximum number of multiplications per sample time is approximately $1.5L + 0.75M + 4$, which represents a savings of approximately 25% over that of the original implementation.

III. THE BLOCK-UPDATE ANC-LMS ALGORITHM

In the filtered–X LMS algorithm, the error signal $\epsilon(n)$ depends on past values of $y(n)$, which in turn depend on past values of $\mathbf{W}(n)$. Since past controller coefficients appear in the error signal, the adaptation performance of this system is worse than that of an algorithm with no such coefficient

Block-Update Filtered–X LMS

Equation	Multiplications	Additions
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\mathbf{X}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\underline{\widetilde{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\mathbf{F}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\epsilon_\mu(n) = \mu(n)\epsilon(n)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) - \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\mathbf{F}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2)$ $-x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$	4	4
$\underline{y}(n) = x_w(n-1) - \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\widetilde{\mathbf{F}}_\epsilon(n-1) = \widetilde{\mathbf{F}}(n-1)(\epsilon_\mu(n-1) + \epsilon_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_\epsilon(n-1) - \widetilde{\mathbf{F}}_\Delta(n)\epsilon_\mu(n) \\ \widetilde{\mathbf{F}}_\epsilon(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)\epsilon_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (two time steps)	$3L+1.5M+7$	$4L+1.5M+8$

Table 1: Equations for the block-update filtered–X LMS algorithm.

delays [8]. A method to effectively remove these delays within the filtered–X LMS updates is described in [2]. In this algorithm, a modified error signal $e(n)$ is calculated as

$$e(n) \;=\; \epsilon(n) - \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1) + \mathbf{F}^T(n)\mathbf{W}(n), \tag{37}$$

where $y(n)$ and $f(n)$ are computed as in (1) and (4), respectively. The coefficients of the controller are then updated as

$$\mathbf{W}(n+1) \;=\; \mathbf{W}(n) - \mu(n)e(n)\mathbf{F}(n)\,. \tag{38}$$

If the coefficients in the secondary path model $\underline{\mathbf{H}}$ are accurate, then this system's adaptation behavior is similar to that of the LMS adaptive filter with input signal $f(n)$ and desired response signal $d(n)$. For this reason, we shall refer to (37)–(38) as the active noise control LMS (ANC-LMS) algorithm. This algorithm requires $3L+2M+1$ multiplies, $3L+2M-2$ additions, and $2L+2M+\max(L, M+1)+3$ memory locations per iteration to implement.

We now derive a block-of-two implementation of the ANC-LMS algorithm that is similar to that of the filtered–X LMS algorithm in Section II. In fact, the equations for $y(n)$, $r_{xf}(n)$, and $f(n)$ are the same as those for the filtered–X LMS algorithm as given in equations (18)-(20), (23)-(24),

and (25)-(30) respectively. Moreover, given $[e(n-1)\ e(n)]^T$, the updates for $\mathbf{W}(n)$ are similar to those in (33)–(36) and are given by

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{F}}_e(n-1) - \tilde{\mathbf{F}}_\Delta(n)\mu(n)e(n) \\ \tilde{\mathbf{F}}_e(n-1) + \tilde{\mathbf{F}}_\Delta(n-1)\mu(n-1)e(n-1) \end{bmatrix}, \tag{39}$$

where

$$\tilde{\mathbf{F}}_e(n) = \tilde{\mathbf{F}}(n)(\mu(n)e(n) + \mu(n+1)e(n+1)). \tag{40}$$

In this algorithm, additional computations are required to compute $e(n)$ in a block fashion. We can express $[e(n-1)\ e(n)]^T$ as

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} - \begin{bmatrix} \underline{\mathbf{Y}}^T(n-2)\underline{\mathbf{H}} \\ \underline{\mathbf{Y}}^T(n-1)\underline{\mathbf{H}} \end{bmatrix} + \begin{bmatrix} \mathbf{F}^T(n-1)\mathbf{W}(n-1) \\ \mathbf{F}^T(n)\mathbf{W}(n) \end{bmatrix}. \tag{41}$$

The second term on the right-hand-side of (41) is of the form in (5), and thus (9) can be used to compute it. Finally, the last term on the right-hand-side of (41) can be computed in a fashion similar to that used for $y(n)$, except that the filtered input signal $f(n)$ is used instead of the input signal $x(n)$ in the calculations. Therefore, the resulting computations for $[e(n-1)\ e(n)]^T$ are

$$\begin{aligned} \begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = & \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} - \begin{bmatrix} y_h(n-2) + \underline{\tilde{\mathbf{Y}}}_\Delta^T(n-2)\underline{\tilde{\mathbf{H}}}_1 \\ y_h(n-2) - \underline{\tilde{\mathbf{Y}}}_\Delta^T(n-1)\underline{\tilde{\mathbf{H}}}_0 \end{bmatrix} \\ & + \begin{bmatrix} f_w(n-1) + \tilde{\mathbf{F}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1) \\ f_w(n-1) - \tilde{\mathbf{F}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) \end{bmatrix} - \begin{bmatrix} 0 \\ \mu(n-1)e(n-1)r_{ff}(n) \end{bmatrix} \end{aligned} \tag{42}$$

where

$$\underline{\tilde{\mathbf{Y}}}_\Delta(n) = \underline{\tilde{\mathbf{Y}}}(n-1) - \underline{\tilde{\mathbf{Y}}}(n) \tag{43}$$

$$y_h(n) = \underline{\tilde{\mathbf{Y}}}^T(n)(\underline{\tilde{\mathbf{H}}}_0 + \underline{\tilde{\mathbf{H}}}_1) \tag{44}$$

$$\underline{\tilde{\mathbf{Y}}}(n) = [y(n)\ y(n-2)\ \cdots\ y(n-M+2)]^T \tag{45}$$

$$f_w(n) = \tilde{\mathbf{F}}^T(n)(\widetilde{\mathbf{W}}_0(n) + \widetilde{\mathbf{W}}_1(n)) \tag{46}$$

$$r_{ff}(n) = r_{ff}(n-2) + \Delta_{ff}(n) - \Delta_{ff}(n-L) \tag{47}$$

$$\Delta_{ff}(n) = f(n)f(n-1) + f(n-1)f(n-2). \tag{48}$$

Note that equation (42) assumes that $e(n-1)$ is computed before $e(n)$, so that $e(n-1)$ can be used in the last term on the right-hand-side of (42).

Table 2 lists the equations and the number of operations needed by the block ANC-LMS algorithm for two time steps. This algorithm requires $4.5L + 3M + 12$ multiplications, $6L + 3M + 17$ additions, and $2.5L+3M+1.5\max(L, M+1)+15$ memory locations for two time steps. With proper partitioning of the operations over successive time samples, it uses a maximum of $2.25L + 1.5M + 6$ multiplications, which represents a savings of roughly 25% over that of the original implementation.

IV. THE BLOCK-UPDATE FAST ANC-LMS ALGORITHM

If the output actuator of the controller is located near the error sensor and the acoustic plant is not overly-resonant, then it may be possible to select the plant model filter length M such that

Block-Update ANC-LMS		
Equation	*Multiplications*	*Additions*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\mathbf{F}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$y_h(n-2) = \widetilde{\mathbf{Y}}^T(n-2)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{Y}}(n-1) = \begin{bmatrix} y(n-1) \\ \widetilde{\mathbf{Y}}(n-3) \end{bmatrix}$	0	0
$f_w(n-1) = \widetilde{\mathbf{F}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$\widetilde{\mathbf{Y}}_\Delta(n-2) = \begin{bmatrix} y(n-3)-y(n-2) \\ \widetilde{\mathbf{Y}}_\Delta(n-4) \end{bmatrix}$	0	1
$e_\mu(n-1) = \mu(n)(\epsilon(n-1) - (y_h(n-2) + \widetilde{\mathbf{Y}}_\Delta^T(n-2)\widetilde{\underline{\mathbf{H}}}_1) + (f_w(n-1) + \widetilde{\mathbf{F}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)))$	$0.5L + 0.5M + 1$	$0.5L + 0.5M + 2$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\mathbf{F}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2) - x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$	4	4
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - e_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\widetilde{\mathbf{Y}}(n) = \begin{bmatrix} y(n) \\ \widetilde{\mathbf{Y}}(n-2) \end{bmatrix}$	0	0
$r_{ff}(n) = r_{ff}(n-2) + f(n)f(n-1) + f(n-1)f(n-2) - f(n-L)f(n-L-1) - f(n-L-1)f(n-L-2)$	4	4
$\widetilde{\mathbf{Y}}_\Delta(n-1) = \begin{bmatrix} y(n-2)-y(n-1) \\ \widetilde{\mathbf{Y}}_\Delta(n-3) \end{bmatrix}$	0	1
$e_\mu(n) = \mu(n)(\epsilon(n) - e_\mu(n-1)r_{ff}(n) - (y_h(n-2) - \widetilde{\mathbf{Y}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_0) + (f_w(n-1) - \widetilde{\mathbf{F}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1)))$	$0.5L + 0.5M + 2$	$0.5L + 0.5M + 3$
$\widetilde{\mathbf{F}}_e(n-1) = \widetilde{\mathbf{F}}(n-1)(e_\mu(n-1) + e_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_e(n-1) - \widetilde{\mathbf{F}}_\Delta(n)e_\mu(n) \\ \widetilde{\mathbf{F}}_e(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)e_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (for two time steps)	$4.5L + 3M + 12$	$6L + 3M + 17$

Table 2: Equations for the block-update ANC-LMS algorithm.

$L > 3M$. In such cases, a more-efficient alternate form of the ANC-LMS algorithm can be used [3]. This algorithm employs an efficient method for calculating $e(n)$ in (37), as given by

$$e(n) = \epsilon(n) + \underline{\mathbf{H}}^T \underline{\mathbf{U}}(n-1) \tag{49}$$

$$\underline{\mathbf{U}}(n) = \begin{bmatrix} 0 \\ \overline{\mathbf{U}}(n-1) \end{bmatrix} - \underline{\mathbf{R}}_{xf}(n)\mu(n)e(n) \tag{50}$$

$$\underline{\mathbf{R}}_{xf}(n) = \underline{\mathbf{R}}_{xf}(n-1) + \underline{\mathbf{X}}(n)f(n) - \underline{\mathbf{X}}(n-L)f(n-L)\,, \tag{51}$$

where $\underline{\mathbf{U}}(n)$ and $\underline{\mathbf{R}}_{xf}(n-1)$ are M–dimensional vectors and $\overline{\mathbf{U}}(n)$ contains the first $M-1$ elements of $\underline{\mathbf{U}}(n)$. All of the other quantities are calculated as in the ANC-LMS algorithm as given by equations (1), (4), and (38). In its original implementation, this algorithm requires $2L + 5M + 1$ multiplications, $2L + 5M - 2$ additions, and $3L + 3M + 4$ memory locations at each sample time.

We now derive a block-of-two implementation of the fast ANC-LMS algorithm. This implementation computes $f(n)$, $y(n)$, and $\mathbf{W}(n)$ in the same fashion as the block-update ANC-LMS algorithm. Only the calculations for $e(n)$ are different and are given by

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} + \begin{bmatrix} u_h(n-1) + (\tilde{\mathbf{U}}_1^T(n-2) - \tilde{\mathbf{U}}_0^T(n-2))\tilde{\mathbf{H}}_1 \\ u_h(n-1) - (\tilde{\mathbf{U}}_0^T(n-2) - \tilde{\mathbf{U}}_{-1}^T(n-2))\tilde{\mathbf{H}}_0 \end{bmatrix} - \begin{bmatrix} 0 \\ r_{ff}(n-1)\mu(n-1)\epsilon(n-1) \end{bmatrix}. \tag{52}$$

where

$$r_{ff}(n-1) = \underline{\mathbf{H}}^T \underline{\mathbf{R}}_{xf}(n-1) \tag{53}$$

$$u_h(n) = \tilde{\mathbf{U}}_0^T(n-1)(\tilde{\mathbf{H}}_0 + \tilde{\mathbf{H}}_1) \tag{54}$$

$$\tilde{\mathbf{U}}_i(n) = [u_i(n)\; u_{i+2}(n)\; \cdots\; u_{i+M-2}(n)]^T\,, \tag{55}$$

$u_i(n)$ is the $(i+1)$th element of $\underline{\mathbf{U}}(n)$, and $u_i(n) = 0$ for $i < 0$. Moreover, the term $r_{ff}(n)$ in (52) can be easily computed as

$$r_{ff}(n) = \sum_{i=0}^{L-1} \underline{\mathbf{H}}^T \underline{\mathbf{X}}(n-i-1)f(n-i-1) = \sum_{i=0}^{L-1} f(n-i)f(n-i-1) \tag{56}$$

$$= r_{ff}(n-1) + f(n)f(n-1) - f(n-L)f(n-L-1). \tag{57}$$

The updates for $\underline{\mathbf{U}}(n)$ and $\underline{\mathbf{R}}_{xf}(n)$ in this block version are the same as in (50) and (51) and are performed twice per block.

Table 3 lists the operations for the block-of-two implementation of the fast ANC-LMS algorithm. The version employs $3L + 9M + 12$ multiplications, $4L + 9.5M + 14$ additions, and $4L + 5M + 16$ memory locations for two time steps. Thus, the algorithm requires approximately $1.5L + 4.5M + 6$ multiplications at each sample time, which represents a savings of $0.5(L + M)$ multiplications per sample time as compared to those of the original implementation.

V. THE BLOCK-UPDATE PERIODIC FILTERED–X LMS ALGORITHM

In some situations, the reductions in complexity provided by the previously-described methods may not be sufficient to allow the chosen algorithm to be implemented on the available hardware platform. In these situations, it is reasonable to consider alternative algorithms that sacrifice some adaptation performance while reducing the implementation complexity of the overall system. In this section, we consider a modification of the filtered-X LMS algorithm in which every Nth error

Block-Update Fast ANC-LMS

Equation	*Multiplications*	*Additions*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\underline{\mathbf{F}}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\underline{\mathbf{F}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$\underline{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \underline{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$u_h(n-1) = \widetilde{\underline{\mathbf{U}}}_0^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$e_\mu(n-1) = \mu(n)(\epsilon(n-2)$		
$+u_h(n-1) + (\widetilde{\underline{\mathbf{U}}}_1^T(n-2) - \widetilde{\underline{\mathbf{U}}}_0^T(n-2))\widetilde{\underline{\mathbf{H}}}_1)$	$0.5M+1$	$M+1$
$\underline{\mathbf{R}}_{xf}(n-1) = \underline{\mathbf{R}}_{xf}(n-2) + \underline{\mathbf{X}}(n-1)f(n-1) - \underline{\mathbf{X}}(n-L-1)f(n-L-1)$	$2M$	$2M$
$\underline{\mathbf{U}}(n-1) = \begin{bmatrix} 0 \\ \underline{\mathbf{U}}(n-2) \end{bmatrix} - \underline{\mathbf{R}}_{xf}(n-1)e_\mu(n-1)$	M	M
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\underline{\mathbf{F}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\underline{\mathbf{F}}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\underline{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \underline{\mathbf{X}}(n-1) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2)$	4	4
$-x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$		
$y(n) = x_w(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$e_\mu(n) = \mu(n)(\epsilon(n) - e_\mu(n-1)r_{ff}(n)$		
$+u_h(n-1) - (\widetilde{\underline{\mathbf{U}}}_0^T(n-2) - \widetilde{\underline{\mathbf{U}}}_{-1}^T(n+2))\widetilde{\underline{\mathbf{H}}}_0$	$0.5M+2$	$0.5M+2$
$\underline{\mathbf{R}}_{fx}(n) = \underline{\mathbf{R}}_{xf}(n-1) + \underline{\mathbf{X}}(n)f(n) - \underline{\mathbf{X}}(n-L)f(n-L)$	$2M$	$2M$
$\underline{\mathbf{U}}(n) = \begin{bmatrix} 0 \\ \underline{\mathbf{U}}(n-1) \end{bmatrix} - \underline{\mathbf{R}}_{xf}(n)e_\mu(n-1)$	M	M
$r_{ff}(n) = r_{ff}(n-2) + f(n)f(n-1) + f(n-1)f(n-2)$		
$-f(n-L)f(n-L-1) - f(n-L-1)f(n-L-2)$	4	4
$\widetilde{\mathbf{F}}_e(n-1) = \widetilde{\mathbf{F}}(n-1)(e_\mu(n-1) + e_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_e(n-1) - \widetilde{\mathbf{F}}_\Delta(n)e_\mu(n) \\ \widetilde{\mathbf{F}}_e(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)e_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (for two time steps)	$3L+9M+12$	$4L+9.5M+14$

Table 3: Equations for the block-update fast ANC-LMS algorithm.

signal $\epsilon(n)$ is used to adjust the coefficients of the system. Termed the periodic filtered-X LMS algorithm [4], the coefficient updates for this system are given by

$$w_i(n+1) = \begin{cases} w_i(n) - \mu(l)\epsilon(l)f(n-i+1) & \text{if } (n+i)\text{mod}N = 0 \\ & \text{and } l = N \bmod(n/N) \\ w_i(n) & \text{otherwise} \end{cases} . \quad (58)$$

This algorithm updates L/N coefficients at each time step; thus, after N time instants, all coefficients have been updated using the same error signal. All other equations for this scheme are the same as that for the filtered-X LMS algorithm. For $N = 2$, this algorithm requires $1.5L + M + 1$ multiplications, $1.5L + M - 2$ additions, and $2L + M + \max(L, M+1) + 3$ memory locations per time step to implement.

We can easily develop a block-of-two implementation of the periodic filtered-X LMS algorithm by noting that the coefficient updates over two time instants are given by

$$\mathbf{W}(n+1) = \mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)\mathbf{F}(n-1). \quad (59)$$

In addition, the correlation term $r_{xf}(n)$ has a two-time-step recursion as

$$r_{xf}(n) = r_{xf}(n-2) + x(n-1)f(n-2) - x(n-L-1)f(n-L-2). \quad (60)$$

The remaining equations are the same as those of the block filtered-X LMS algorithm of Section II, except that the calculations for $\tilde{\mathbf{F}}_\Delta(n-i)$ and $\tilde{\mathbf{F}}_\epsilon(n-i)$ are not required.

Table 4 lists the equations and number of operations for this algorithm. The algorithm requires $2.5L + 1.5M + 4$ multiplies and $3L + 1.5M + 3$ additions per two time steps and uses $3L + 1.5M + 1.5\max(L, M+1) + 3$ memory locations. With proper partitioning of the operations across each sample time, the maximum number of multiplications required is $1.25L + 0.75M + 2$. which represents a savings of $0.25(L+M)$ multiplies over the original implementation.

VI. THE BLOCK-UPDATE SEQUENTIAL FILTERED–X LMS ALGORITHM

While the periodic filtered-X LMS algorithm described in the last section updates only L/N coefficients at each iteration, it requires the calculation of every filtered input signal $f(n)$. In [4], an alternative algorithm is described that uses every Nth value of the filtered input sequence, which saves computations in active noise control applications. Termed the sequential filtered-X LMS algorithm, the coefficient updates are given by

$$w_i(n+1) = \begin{cases} w_i(n) - \mu(n)\epsilon(n)f(n-i+1) & \text{if } (n-i+1)\text{mod}N = 0 \\ w_i(n) & \text{otherwise} \end{cases} . \quad (61)$$

When $N = 2$, these updates can be conveniently represented as

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \mathbf{W}_1(n-1) \end{bmatrix} - \begin{bmatrix} \mu(n-1)\epsilon(n-1)\tilde{\mathbf{F}}(n-1) \\ \mu(n)\epsilon(n)\tilde{\mathbf{F}}(n-1) \end{bmatrix} . \quad (62)$$

and the output signal is computed using (1). This algorithm requires $1.5L + 0.5M + 1$ multiplications and $1.5L + 0.5M - 1$ additions at each time step on average. It uses $2L + 0.5M + \max(L, M+1) + 3$ memory locations. As is shown in [4] for noise cancellation tasks, the average performances of the sequential and periodic LMS algorithms are nearly identical for the same coefficient update rates.

We now state the block-update version of the sequential filtered-X LMS algorithm. In this case, the fast convolution technique is only applied to the input-output controller calculation, and the

Block-Update Periodic Filtered–X LMS		
Equation	*Multiplications*	*Additions*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) - \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n-1)f(n-2) - x(n-L-1)f(n-L-2)$	2	2
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}(n-1) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix} \epsilon_\mu(n-1)$	L	L
Totals (two time steps)	$2.5L + 1.5M + 4$	$3L + 1.5M + 3$

Table 4: Equations for the block-update periodic filtered–X LMS algorithm.

Block-Update Sequential Filtered–X LMS		
Equation	*Multiplications*	*Additions*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\underline{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-2) \\ \underline{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$f(n-1) = \underline{\mathbf{X}}^T(n-1)\underline{\mathbf{H}}$	M	$M-1$
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\epsilon_\mu(n) = \mu(n)\epsilon(n)$	1	0
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	0	1
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) - x(n-L)f(n-L-1)$	2	2
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}(n-1)\epsilon_\mu(n-1) \\ \widetilde{\mathbf{F}}(n-1)\epsilon_\mu(n) \end{bmatrix}$	L	L
Totals (two time steps)	$2.5L + M + 5$	$3L + M + 3$

Table 5: Equations for the block-update sequential filtered–X LMS algorithm.

Comparison of Original and Block-Update Algorithms resource usage per time step.			
Algorithm	*Multiplies*	*Additions*	*Memory*
FXLMS	$2L + M + 1$	$2L + M - 2$	$2L + M + \max(L, M + 1) + 3$
Block FXLMS	$1.5L + 0.75M + 4$	$2L + 0.75M + 4$	$2.5L + 1.5M + 1.5\max(L, M + 1) + 14$
ANC-LMS	$3L + 2M + 1$	$3L + 2M - 2$	$2L + 2M + \max(L, M + 1) + 3$
Block ANC-LMS	$2.25L + 1.5M + 6$	$3L + 1.5M + 9$	$2.5L + 3M + 1.5\max(L, M + 1) + 15$
Fast ANC-LMS	$2L + 5M + 1$	$2L + 5M - 2$	$3L + 3M + 4$
Block Fast ANC-LMS	$1.5L + 4.5M + 6$	$2L + 4.75M + 7$	$4L + 5M + 16$
Periodic FXLMS	$1.5L + M + 1$	$1.5L + M - 2$	$2L + M + \max(L, M + 1) + 3$
Block Periodic FXLMS	$1.25L + 0.75M + 3$	$1.5L + 0.75M + 4$	$3L + 1.5M + 1.5\max(L, M + 1) + 3$
Sequential FXLMS	$1.5L + 0.5M + 1$	$1.5L + 0.5M - 1$	$2L + M + \max(L, M + 1) + 3$
Block Sequential FXLMS	$1.25L + 0.5M + 3$	$1.5L + 0.5M + 2$	$2L + 1.5M + 1.5\max(L, M + 1) + 11$

Table 6: Complexity comparison of standard and block-update algorithms.

calculations associated with computing $f(n)$ for even time instants can be removed. Table 5 shows the operations for this version of the algorithm. This method requires $2.5L + M + 5$ multiplications, $3L + M + 3$ additions, and $2L + 1.5M + 1.5\max(L, M + 1) + 11$ memory locations to implement for two time steps. Thus, the average number of multiplies per time step is $1.25L + 0.5M + 3$, a savings of approximately $0.25L$ multiplications over the original implementation.

VII. SUMMARY AND CONCLUSIONS

In summary, we have derived new implementations of several existing algorithms for feedforward active noise control. Table 6 summarizes the number of multiplications additions, and memory required for the standard and block-of-two implementations of the algorithms. Our results indicate that the fast convolution technique is effective at reducing the complexities of these algorithms, and up to a 25% reduction in multiplications can be achieved in some cases.

ACKNOWLEDGEMENT

This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

REFERENCES

[1] "On adaptive inverse control," B. Widrow, D. Shur, and S. Shaffer, *Proc. 15th Asilomar Conf.*, Pacific Grove, CA, 185-189 (1981)

[2] "Active noise cancellation using a modified form of the filtered–X LMS algorithm," E. Bjarnason, *Proc. EUSIPCO-92*, Brussels, Belgium, 1053-1056 (1992)

[3] "Fast, exact filtered–X LMS and LMS algorithms for multichannel active noise control," S.C. Douglas, *Proc. ICASSP-97*, Munich, Germany, 399-402 (1997)

[4] "Adaptive filters employing partial updates," S.C. Douglas, *IEEE Trans. Circ. Syst. II: Anal. Dig. Sig. Proc.*, **44**, 209-216 (1997)

[5] "Time- and frequency-domain X-block least-mean-square algorithms for active noise control," Q. Shen and A.S. Spanias, *Noise Contr. Eng. J.*, **44**, 281-293 (1996)

[6] "Short-length FIR filters and their use in nonrecursive filtering," Z.J. Mou and P. Duhamel, *IEEE Trans. Sig. Proc.*, **39**, 1322-1332 (1991).

[7] "A fast exact least mean square adaptive algorithm," J. Benesty and P. Duhamel, *IEEE. Trans. Sig. Proc.*, **40**, 2904-2912 (1992).

[8] "Analysis of the filtered–X LMS algorithm," E. Bjarnason, *IEEE Trans. Speech Audio Proc.*, **3**, 504-514 (1995).

Fast Exact Adaptive Algorithms for Feedforward Active Noise Control

David S. Nelson[1], Scott C. Douglas[2] and Marc Bodson[3] *

[1]Communications Research Laboratory
L3 Communications Corp.
Salt Lake City, Utah 84120 USA

[2]Department of Electrical Engineering
Southern Methodist University
Dallas, Texas 75275 USA

[3]Department of Electrical Engineering
University of Utah
Salt Lake City, Utah 84112 USA

Abstract– **The fast exact least-mean-square (LMS) algorithm is a computationally-efficient method for computing the outputs and updates for an adaptive LMS finite-impulse-response (FIR) filter. In this paper, we extend this method to several useful algorithms for feedforward active noise control: the filtered-X LMS, modified filtered-X LMS, efficient modified filtered-X LMS, periodic filtered-X LMS, and sequential filtered-X LMS algorithms, respectively. Choosing a block size of two produces overall behaviors for these fast exact versions that are identical to their non-block counterparts while reducing the numbers of multiplies by up to 25% over those required by the standard algorithms. We then describe Motorola DSP96002 DSP-based implementations of the standard and fast exact versions of the filtered-X LMS algorithm. Our results show that the fast exact implementation can allow a 27.4% increase in the filter lengths over those of the standard implementation on this processor, which is close to the 33.3% increase that would be expected if the number of multiplies were a true indication of an algorithm's complexity.**

submitted to

International Journal of Adaptive Control and Signal Processing

Robert W. Stewart and Stephan Weiss, Special Editors

*Please address correspondence to: Scott C. Douglas, Department of Electrical Engineering, School of Engineering and Applied Science, Southern Methodist University, P.O. Box 750338, Dallas, TX 75275 USA. Voice: (214) 768-3113. FAX: (214) 768-3573. Electronic mail address: douglas@seas.smu.edu. World Wide Web URL: http://www.seas.smu.edu/ee/.

1 Introduction

In active noise control (ANC), unwanted acoustic noise caused by a primary source is attenuated by injecting via a secondary source an acoustic field that that is equal in magnitude and opposite in phase to that of the unwanted noise in a physical region of interest. The two acoustic fields destructively interfere, thereby reducing the level of the unwanted acoustic noise. Due to the advances in digital signal processing (DSP) hardware, ANC has become practical for many useful applications [1]–[10]. As more-advanced hardware and more-efficient algorithms are developed, ANC will likely become a common method for attenuating unwanted acoustic noise.

Figure 1 shows a block diagram of a single-channel feedforward ANC system. An input signal $x(n)$ is measured and processed by a time-varying finite-impulse-response (FIR) filter as

$$y(n) \;=\; \sum_{l=0}^{L-1} w_l(n)x(n-l) \;=\; \mathbf{X}^T(n)\mathbf{W}(n), \tag{1}$$

where the vector $\mathbf{X}(n) = [x(n) \;\cdots\; x(n-L+1)]^T$ contains the L most-recent samples of the input signal and $\mathbf{W}(n) = [w_0(n) \;\cdots\; w_{L-1}(n)]^T$ is the coefficient vector of the controller at time n. The output signal of the controller propagates to the desired quiet region, where it combines with the primary signal to be canceled, denoted by $d(n)$. An error sensor measures the linear combination of these two signals as modeled by the equation

$$\epsilon(n) \;=\; d(n) + \underline{\mathbf{H}}^T\underline{\mathbf{Y}}(n-1), \tag{2}$$

where $\epsilon(n)$ is the measured error signal, $\underline{\mathbf{Y}}(n-1) = [y(n-1) \;\cdots\; y(n-M)]^T$ and where the FIR filter with impulse response $\underline{\mathbf{H}} = [h_1 \;\cdots\; h_M]^T$ is assumed to accurately represent the transfer characteristics of the output-actuator-to-error-sensor secondary path. The task is to adjust $\mathbf{W}(n)$ so that the magnitude of $\epsilon(n)$ is reduced over time.

Various algorithms for adjusting $\mathbf{W}(n)$ for ANC have been proposed [11]–[21]. For the most part, they involve multiply and add operations that are similar to the FIR filter output calculation, and they typically employ an estimate of the secondary path impulse response in $\underline{\mathbf{H}}$. Although multiply and add operations are simple to implement in typical real-time DSP hardware, it is often desirable to minimize the arithmetic complexities of these algorithms 1) to allow the algorithm to be implemented on a less-expensive processor platform; 2) to reduce the demands on a particular processor platform so that other operations can be simultaneously performed by the system; or 3) to obtain higher cancellation performance in broadband noise control applications on a given processing platform by increasing the lengths of $\mathbf{W}(n)$ and $\mathbf{H}$ so that the controller can more accurately represent the physical paths within the system.

Recently, a fast exact LMS algorithm was introduced that exactly-computes the outputs of the LMS adaptive filter while reducing the overall complexity of the system via block coefficient updates [23]. The method reduces the number of multiplies required by the system by 25% for a block length of two while keeping the number of adds approximately the same, and greater savings can be obtained for larger block lengths.

In this paper, we extend the fast convolution and updating methods employed within the fast exact LMS algorithm [22, 23] to five different single-channel ANC algorithms: the filtered-X LMS [11, 12], modified filtered-X LMS [13, 14, 15], efficient modified filtered-X LMS [20], periodic filtered-X LMS [19], and sequential filtered-X LMS [19] algorithms, respectively. Choosing a block length of two guarantees that the input-output characteristics of the original implementations are exactly maintained in our new implementations, unlike other block-update methods that introduce significant delays into the coefficient updates [17]. We show that the numbers of multiplies required for the new implementations are reduced by as much as 25% over their original implementations, and the numbers of adds required by the algorithms are also reduced. We then describe a real-time implementation of the fast exact filtered-X LMS algorithm on the Motorola DSP96002 processor. Real-time experiments indicate that the new implementation provides mathematically-equivalent behavior to that of the standard implementation. Because of its computational efficiency, however, the new implementation allows the system filter lengths to be increased by as much as 27.4% over those of the original implementation without failure, a factor close to the 33.3% increase suggested if the number of multiplies were a true indication of an algorithm's complexity.

This paper is organized as follows. In the next section, we describe fast exact methods for convolution and the coefficient updating of an LMS adaptive filter. Section 3 provides extensions of these methods to the five different ANC algorithms mentioned previously and compares their computational complexities to those of their original implementations. We discuss an implementation of the fast exact filtered-X LMS algorithm on the Motorola DSP96002 processor in Section 4, and real-time experiments are described. Section 5 presents our conclusions.

To aid the reader, Table 1 lists the notation used in this paper. In most cases, the notation follows similar conventions used in the presentation of other adaptive algorithms in the literature.

2 Review of Fast Exact Methods

Before presenting the new algorithms, we first review previously-derived methods for the fast exact implementation of constant-coefficient and gradient adaptive FIR filters [22, 23]. Our pre-

sentation uses a generalized notation that allows us to apply the methods to other algorithms in Section 3. Here, we focus on a block-of-two implementation, in which two successive outputs or updates are computed in parallel, as this implementation allows for mathematically-exact zero-latency real-time controller implementations in the algorithms of Section 3.

2.1 Fixed-Coefficient Fast FIR Convolution

In [22], a simplified fixed-coefficient FIR filter implementation is described that calculates terms that can be reused at successive time instants to calculate the filter output signal via low-complexity correction terms. Two successive outputs from an L-coefficient FIR filter with input signal $x(n)$ and coefficient vector $\mathbf{W} = [w_0 \ \cdots \ w_{L-1}]^T$ can be computed as

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \mathbf{W} = \begin{bmatrix} \tilde{\mathbf{X}}^T(n-1) & \tilde{\mathbf{X}}^T(n-2) \\ \tilde{\mathbf{X}}^T(n) & \tilde{\mathbf{X}}^T(n-1) \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{W}}_0 \\ \widetilde{\mathbf{W}}_1 \end{bmatrix}, \tag{3}$$

where

$$\tilde{\mathbf{X}}(n) = [x(n) \ x(n-2) \ \cdots \ x(n-L+2)]^T \tag{4}$$

$$\widetilde{\mathbf{W}}_i = [w_i \ w_{2+i} \ \cdots \ w_{L-2+i}]^T . \tag{5}$$

Expanding the last term on the RHS of (3) and adding and subtracting the term $\tilde{\mathbf{X}}^T(n-1)\widetilde{\mathbf{W}}_1$ from both entries gives

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} x_w(n-1) + \tilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1 \\ x_w(n-1) - \tilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0 \end{bmatrix}, \tag{6}$$

where we have defined

$$x_w(n) = \tilde{\mathbf{X}}^T(n)(\widetilde{\mathbf{W}}_0 + \widetilde{\mathbf{W}}_1) \tag{7}$$

$$\tilde{\mathbf{X}}_\Delta(n) = \tilde{\mathbf{X}}(n-1) - \tilde{\mathbf{X}}(n) . \tag{8}$$

Equations (6) and (7) constitute the fast FIR convolution method. This method requires $1.5L$ multiplies for every two time steps, since $x_w(n)$ need only be computed once. Thus, this implementation uses $0.75L$ multiplies per time step on average, a savings of $0.25L$ multiplies over the standard FIR filter implementation. Moreover, storing the difference terms $x(n-i)-x(n-i-1)$ in $\tilde{\mathbf{X}}_\Delta(n)$ reduces the number of adds needed to compute the filter output.

The above method can be extended to compute a block of N output signals. In active noise control, however, it is important not to introduce any output latency, which typically reduces the performance of the controller for broadband noise sources. Since all values used to compute $y(n-1)$ are available at time $n-1$ in the above method, it may be implemented in a sequential way without introducing any output latency. For this reason, we focus on the block-of-two case in this paper.

2.2 Fast Convolution for Gradient Adaptive Filters

When $\mathbf{W}(n)$ is time-varying, the coefficient updates must be included within the above calculations. For all of the algorithms considered in this paper, the update for $\mathbf{W}(n)$ has the form

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \delta(n)\mathbf{\Delta}(n), \tag{9}$$

where $\delta(n)$ is an error signal and $\mathbf{\Delta}(n)$ is a regressor vector. Write (1) and (9) at time $n-1$ as

$$\mathbf{W}(n) = \mathbf{W}(n-1) - \delta(n-1)\mathbf{\Delta}(n-1) \tag{10}$$

$$y(n-1) = \mathbf{X}^T(n-1)\mathbf{W}(n-1). \tag{11}$$

Substituting $\mathbf{W}(n)$ in (10) into (1), we obtain

$$y(n) = \mathbf{X}^T(n)\mathbf{W}(n-1) - \delta(n-1)r_{x\Delta}(n), \tag{12}$$

where we have defined

$$r_{x\Delta}(n) = \mathbf{X}^T(n)\mathbf{\Delta}(n-1)\,. \tag{13}$$

Considering two successive output samples, we can write

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \mathbf{W}(n-1) - \begin{bmatrix} 0 \\ \delta(n-1)r_{x\Delta}(n) \end{bmatrix}. \tag{14}$$

The first term on the RHS of (14) has the same form as that in (3), and as such, (6)–(7) can be used to compute it with $\mathbf{W} = \mathbf{W}(n-1)$. In addition, the sliding-window nature of $r_{x\Delta}(n)$ for gradient-based updates can be used to simplify its calculation, as will be shown.

2.3 Fast Updating of Gradient Adaptive FIR Filters

In [23], a method for calculating the outputs and updates for the LMS adaptive filter in a block fashion is described. Here, we generalize this result. Two successive iterations of (9) are written as

$$\mathbf{W}(n+1) = (\mathbf{W}(n-1) - \delta(n-1)\mathbf{\Delta}(n-1)) - \delta(n)\mathbf{\Delta}(n), \tag{15}$$

or, equivalently,

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{\Delta}}_\delta(n-1) - \widetilde{\mathbf{\Delta}}_\Delta(n)\delta(n) \\ \widetilde{\mathbf{\Delta}}_\delta(n-1) + \widetilde{\mathbf{\Delta}}_\Delta(n-1)\delta(n-1) \end{bmatrix}, \tag{16}$$

where we have defined

$$\widetilde{\mathbf{\Delta}}_\Delta(n) = \widetilde{\mathbf{\Delta}}(n-1) - \widetilde{\mathbf{\Delta}}(n) \tag{17}$$

$$\widetilde{\mathbf{\Delta}}_\delta(n) = \widetilde{\mathbf{\Delta}}(n)(\delta(n) + \delta(n+1))\,, \tag{18}$$

and $\widetilde{\mathbf{\Delta}}(n)$ contains every other sample of the regressor vector $\Delta(n)$. As the $L/2$-element vector $\widetilde{\mathbf{\Delta}}_\delta(n)$ is common to both block update rows in (16), it need only be computed once. Thus, (16)–(18) requires about $1.5L$ multiplies per two time steps, or about $0.75L$ multiplies on average at each time step, a savings of $0.25L$ multiplies over the standard implementation.

In the next section, we apply these general methods to five different ANC algorithms.

3 Fast Exact ANC Algorithms

3.1 Filtered-X LMS Algorithm

The most popular algorithm for active noise control systems is the filtered-X LMS algorithm shown in Figure 2. In this system, the shaded block represents the physical transfer function from the output actuator to the error sensor, and the remaining blocks represent the computations performed by the controller. The coefficient updates are given by

$$\mathbf{W}(n+1) \;=\; \mathbf{W}(n) - \mu(n)\epsilon(n)\mathbf{F}(n)\,, \tag{19}$$

where $\mathbf{W}(n)$ and $\epsilon(n)$ are defined as previously, $\mathbf{F}(n) = [f(n) \;\cdots\; f(n-L+1)]^T$ contains the L most-recent samples of the filtered input signal, and $\mu(n)$ is the step size at time n. In this system, $f(n)$ is computed as

$$f(n) = \underline{\mathbf{H}}^T\underline{\mathbf{X}}(n-1), \tag{20}$$

where $\underline{\mathbf{X}}(n-1) = [x(n-1) \;\cdots\; x(n-M)]^T$ contains M samples of the input signal. The output of the controller is computed as in (1). Considering (1), (19), and (20), we find that this algorithm requires $2L+M+1$ multiplies and $2L+M-2$ adds per time step, and it employs $2L+M+\max(L, M+1)+3$ memory locations within the controller.

We now extend the fast methods of Section 2 to derive a fast exact implementation of the filtered-X LMS algorithm. First, we notice that the equation for computing $y(n)$ is identical in form to that in (12). Thus, (14) becomes

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} \;=\; \begin{bmatrix} x_w(n-1) + \tilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1) \\ x_w(n-1) - \tilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) \end{bmatrix} - \begin{bmatrix} 0 \\ \mu(n-1)\epsilon(n-1)r_{xf}(n) \end{bmatrix}, \tag{21}$$

with

$$\widetilde{\mathbf{W}}_i(n) \;=\; [w_i(n)\; w_{2+i}(n) \;\cdots\; w_{L-2+i}(n)]^T \tag{22}$$

$$x_w(n) \;=\; \tilde{\mathbf{X}}^T(n)(\widetilde{\mathbf{W}}_0(n) + \widetilde{\mathbf{W}}_1(n)) \tag{23}$$

$$r_{xf}(n) \;=\; \mathbf{X}^T(n)\mathbf{F}(n-1)\,. \tag{24}$$

To compute $r_{xf}(n)$ in an efficient manner, we note that

$$r_{xf}(n) = \sum_{i=0}^{L} x(n-i)f(n-i-1) \tag{25}$$

$$= r_{xf}(n-2) + \Delta_{xf}(n) - \Delta_{xf}(n-L)\,, \tag{26}$$

where $\Delta_{xf}(n) = x(n)f(n-1) + x(n-1)f(n-2)$.

The computation of $[f(n-1)\ f(n)]^T$ is similar to the original fast convolution method in (6), as $\underline{\mathbf{H}}$ is fixed. The resulting equations are

$$\begin{bmatrix} f(n-1) \\ f(n) \end{bmatrix} = \begin{bmatrix} x_h(n-1) + \tilde{\underline{\mathbf{X}}}_{\Delta}^T(n-1)\tilde{\underline{\mathbf{H}}}_1 \\ x_h(n-1) - \tilde{\underline{\mathbf{X}}}_{\Delta}^T(n)\tilde{\underline{\mathbf{H}}}_0 \end{bmatrix}, \tag{27}$$

where we define

$$\tilde{\underline{\mathbf{X}}}_{\Delta}(n) = \tilde{\underline{\mathbf{X}}}(n-1) - \tilde{\underline{\mathbf{X}}}(n) \tag{28}$$

$$\tilde{\underline{\mathbf{H}}}_i = [h_{1+i}\ h_{3+i}\ \cdots\ h_{M-1+i}]^T \tag{29}$$

$$\tilde{\underline{\mathbf{X}}}(n-1) = [x(n-1)\ x(n-3)\ \cdots\ x(n-M+1)]^T \tag{30}$$

$$x_h(n) = \tilde{\underline{\mathbf{X}}}^T(n)\tilde{\underline{\mathbf{H}}}_{\Delta} \tag{31}$$

$$\tilde{\underline{\mathbf{H}}}_{\Delta} = \tilde{\underline{\mathbf{H}}}_0 + \tilde{\underline{\mathbf{H}}}_1\,. \tag{32}$$

A similar method could be used to compute $[\epsilon(n-1)\ \epsilon(n)]^T$ for simulation purposes; in practice, however, $\epsilon(n)$ is measured by the error sensor and need not be computed.

To update the filter coefficients, we use a similar version of (16) as given by

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{F}}_{\epsilon}(n-1) - \tilde{\mathbf{F}}_{\Delta}(n)\mu(n)\epsilon(n) \\ \tilde{\mathbf{F}}_{\epsilon}(n-1) + \tilde{\mathbf{F}}_{\Delta}(n-1)\mu(n-1)\epsilon(n-1) \end{bmatrix}, \tag{33}$$

where we have defined

$$\tilde{\mathbf{F}}_{\Delta}(n) = \tilde{\mathbf{F}}(n-1) - \tilde{\mathbf{F}}(n) \tag{34}$$

$$\tilde{\mathbf{F}}(n) = [f(n)\ f(n-2)\ \cdots\ f(n-L+2)]^T \tag{35}$$

$$\tilde{\mathbf{F}}_{\epsilon}(n) = \tilde{\mathbf{F}}(n)(\mu(n)\epsilon(n) + \mu(n+1)\epsilon(n+1))\,. \tag{36}$$

Table 2 list the equations for the fast exact implementation of the filtered-X LMS algorithm along with the number of multiplies and adds needed for each step. In the table, a horizontal line separates the operations necessary at each time step; however, it is possible to shift some of the calculations from the second half of the block to the first half to balance the computational load of the algorithm at each sample time. This algorithm requires $3L+1.5M+7$ multiplies, $4L+1.5M+8$

adds, and $2.5L + 1.5M + 1.5\max(L, M+1) + 14$ memory locations per two sample times. Thus, the maximum number of multiplies per sample time is approximately $1.5L + 0.75M + 4$, which represents a savings of approximately 25% over that of the original implementation.

3.2 Modified Filtered-X LMS Algorithm

One drawback of the filtered-X LMS algorithm is that the error signal $\epsilon(n)$ depends on past values of $y(n)$, which in turn depend on past values of $\mathbf{W}(n)$. Since past controller coefficients appear in the error signal, the adaptation performance of this system is worse than that of an algorithm with no such coefficient delays [16]. A method to effectively remove these delays within the filtered-X LMS updates is described in [13]. This modified filtered-X LMS algorithm is shown in Figure 3, in which a modified error signal $e(n)$ is calculated as

$$e(n) = \epsilon(n) - \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1) + \mathbf{F}^T(n)\mathbf{W}(n), \tag{37}$$

where $y(n)$ and $f(n)$ are computed as in (1) and (20), respectively. The coefficients of the controller are then updated using the LMS algorithm as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu(n)e(n)\mathbf{F}(n)\,. \tag{38}$$

If the coefficients in the secondary path model $\underline{\mathbf{H}}$ are accurate, then this system's adaptation behavior is similar to that of an LMS adaptive filter with input signal $f(n)$ and desired response signal $d(n)$. The complete algorithm requires $3L + 2M + 1$ multiplies, $3L + 2M - 2$ adds, and $2L + 2M + max(L, M+1) + 3$ memory locations per iteration to implement.

We now derive a fast exact implementation of the fast modified filtered-X LMS algorithm that is similar to that of the filtered-X LMS algorithm in Section 3.1. In fact, the equations for $y(n)$, $r_{xf}(n)$, and $f(n)$ are the same as those for the filtered-X LMS algorithm as given in equations (21)-(23), (26), and (27)-(32) respectively. Moreover, given $[e(n-1)\ e(n)]^T$, the updates for $\mathbf{W}(n)$ are similar to those in (33)–(36) and are given by

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_e(n-1) - \widetilde{\mathbf{F}}_\Delta(n)\mu(n)e(n) \\ \widetilde{\mathbf{F}}_e(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)\mu(n-1)e(n-1) \end{bmatrix} \tag{39}$$

where $\widetilde{\mathbf{F}}_e(n) = \widetilde{\mathbf{F}}(n)(\mu(n)e(n) + \mu(n+1)e(n+1))$.

In this algorithm, additional computations are required to compute $e(n)$ in a block fashion. We can express $[e(n-1)\ e(n)]^T$ as

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} - \begin{bmatrix} \underline{\mathbf{Y}}^T(n-2)\underline{\mathbf{H}} \\ \underline{\mathbf{Y}}^T(n-1)\underline{\mathbf{H}} \end{bmatrix} + \begin{bmatrix} \mathbf{F}^T(n-1)\mathbf{W}(n-1) \\ \mathbf{F}^T(n)\mathbf{W}(n) \end{bmatrix}. \tag{40}$$

The second term on the RHS of (40) is similar to (3), and thus it can be put in a form similar to (6). Finally, the last term on the RHS of (40) can be computed in a fashion similar to that used to compute $y(n)$, except that the filtered input signal $f(n)$ is used instead of the input signal $x(n)$ in the calculations. Thus, the resulting computations for $[e(n-1)\ e(n)]^T$ are

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} - \begin{bmatrix} y_h(n-2) + \tilde{\underline{\mathbf{Y}}}_{\Delta}^T(n-2)\tilde{\underline{\mathbf{H}}}_1 \\ y_h(n-2) - \tilde{\underline{\mathbf{Y}}}_{\Delta}^T(n-1)\tilde{\underline{\mathbf{H}}}_0 \end{bmatrix} + \begin{bmatrix} f_w(n-1) + \tilde{\mathbf{F}}_{\Delta}^T(n-1)\widetilde{\mathbf{W}}_1(n-1) \\ f_w(n-1) - \tilde{\mathbf{F}}_{\Delta}^T(n)\widetilde{\mathbf{W}}_0(n-1) \end{bmatrix} - \begin{bmatrix} 0 \\ \mu(n-1)e(n-1)r_{ff}(n) \end{bmatrix} \quad (41)$$

where

$$\tilde{\underline{\mathbf{Y}}}_{\Delta}(n) = \tilde{\underline{\mathbf{Y}}}(n-1) - \tilde{\underline{\mathbf{Y}}}(n) \quad (42)$$

$$y_h(n) = \tilde{\underline{\mathbf{Y}}}^T(n)(\tilde{\underline{\mathbf{H}}}_0 + \tilde{\underline{\mathbf{H}}}_1) \quad (43)$$

$$\tilde{\underline{\mathbf{Y}}}(n) = [y(n)\ y(n-2)\ \cdots\ y(n-M+2]^T \quad (44)$$

$$f_w(n) = \tilde{\mathbf{F}}^T(n)(\widetilde{\mathbf{W}}_0(n) + \widetilde{\mathbf{W}}_1(n)) \quad (45)$$

$$r_{ff}(n) = r_{ff}(n-2) + \Delta_{ff}(n) - \Delta_{ff}(n-L) \quad (46)$$

$$\Delta_{ff}(n) = f(n)f(n-1) + f(n-1)f(n-2)\,. \quad (47)$$

Note that equation (41) assumes that $e(n-1)$ is computed before $e(n)$, so $e(n-1)$ can be used in the last term on the RHS of (41).

Table 3 lists the equations and the number of operations needed by the fast modified filtered-X LMS algorithm for two time steps. This algorithm requires $4.5L+3M+12$ multiplies, $6L+3M+17$ adds, and $2.5L + 3M + 1.5\max(L, M+1) + 15$ memory locations for two time steps. With proper partitioning of the operations over successive time samples, it uses at most $2.25L + 1.5M + 6$ multiplies at each time step, which represents a savings of roughly 25% over that of the original implementation.

3.3 Efficient Modified Filtered-X LMS Algorithm

If the output actuator of the controller is located near the error sensor and the acoustic plant is not overly-resonant, then the filter length M of the secondary path impulse response $\underline{\mathbf{H}}$ can be chosen to be much less than the filter length L of the controller. In situations where $L > 3M$, the modified filtered-X LMS algorithm can be placed in an alternate form that is more computationally-efficient than that of the original [20]. This efficient modified filtered-X LMS algorithm employs a different method for calculating $e(n)$ in (37), as given by

$$e(n) = \epsilon(n) + \underline{\mathbf{H}}^T\underline{\mathbf{U}}(n-1) \quad (48)$$

$$\underline{\mathbf{U}}(n) = \begin{bmatrix} 0 \\ \overline{\mathbf{U}}(n-1) \end{bmatrix} - \underline{\mathbf{R}}_{xf}(n)\mu(n)e(n) \tag{49}$$

$$\underline{\mathbf{R}}_{xf}(n) = \underline{\mathbf{R}}_{xf}(n-1) + \underline{\mathbf{X}}(n)f(n) - \underline{\mathbf{X}}(n-L)f(n-L)\,, \tag{50}$$

where $\underline{\mathbf{U}}(n)$ and $\underline{\mathbf{R}}_{xf}(n-1)$ are M–dimensional vectors and $\overline{\mathbf{U}}(n-1)$ contains the first $M-1$ elements of $\underline{\mathbf{U}}(n-1)$. All other quantities are calculated as in the original modified filtered-X LMS algorithm in (1), (20), and (38). This algorithm requires $2L+5M+1$ multiplies, $2L+5M-2$ adds, and $3L+3M+4$ memory locations at each sample time to implement.

We now derive a fast exact implementation of the efficient modified filtered-X LMS algorithm. This implementation computes $f(n)$, $y(n)$, and $\mathbf{W}(n)$ as in the fast modified filtered-X LMS algorithm. Only the calculations for $e(n)$ are different and are given by

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} \epsilon(n-1) \\ \epsilon(n) \end{bmatrix} + \begin{bmatrix} u_h(n-1) + (\tilde{\mathbf{U}}_1^T(n-2) - \tilde{\mathbf{U}}_0^T(n-2))\tilde{\mathbf{H}}_1 \\ u_h(n-1) - (\tilde{\mathbf{U}}_0^T(n-2) - \tilde{\mathbf{U}}_{-1}^T(n-2))\tilde{\mathbf{H}}_0 \end{bmatrix} - \begin{bmatrix} 0 \\ r_{ff}(n-1)\mu(n-1)e(n-1) \end{bmatrix}, \tag{51}$$

where

$$r_{ff}(n-1) = \underline{\mathbf{H}}^T\underline{\mathbf{R}}_{xf}(n-1) \tag{52}$$

$$u_h(n) = \tilde{\mathbf{U}}_0^T(n-1)(\tilde{\mathbf{H}}_0 + \tilde{\mathbf{H}}_1) \tag{53}$$

$$\tilde{\mathbf{U}}_i(n) = [u_i(n)\; u_{i+2}(n)\; \cdots\; u_{i+M-2}(n)]^T\,, \tag{54}$$

$u_i(n)$ is the $(i+1)$th element of $\underline{\mathbf{U}}(n)$, and $u_i(n) = 0$ for $i < 0$. Moreover, $r_{ff}(n)$ is computed as

$$r_{ff}(n) = \sum_{i=0}^{L-1} \mathbf{H}^T\underline{\mathbf{X}}(n-i-1)f(n-i-1) = \sum_{i=0}^{L-1} f(n-i)f(n-i-1) \tag{55}$$

$$r_{ff}(n) = r_{ff}(n-1) + f(n)f(n-1) - f(n-L)f(n-L-1). \tag{56}$$

The updates for $\underline{\mathbf{U}}(n)$ and $\underline{\mathbf{R}}_{xf}(n)$ in this fast exact version are the same as in (49) and (50) and are performed twice per block. Table 4 lists the operations for the fast exact implementation of the efficient modified filtered-X LMS algorithm. The version employs $3L+9M+12$ multiplies, $4L+9.5M+14$ adds, and $4L+5M+16$ memory locations for two time steps. Thus, the algorithm requires approximately $1.5L+4.5M+6$ multiplies at each sample time, which represents a savings of $0.5(L+M)$ multiplies per sample time as compared to that of the original implementation.

3.4 Periodic Filtered-X LMS Algorithm

In some situations, the reductions in complexity provided by the previously-described methods may not be sufficient to allow the chosen algorithm to be implemented on a given hardware platform. In

these situations, it is reasonable to consider alternative algorithms that sacrifice some adaptation performance while reducing the system's implementation complexity. The periodic filtered-X LMS algorithm is one such algorithm, in which every Nth error signal $\epsilon(n)$ is used to adjust the controller coefficients [19]. The block diagram for this algorithm is the same as that in Figure 2, and the coefficient updates are

$$w_i(n+1) = \begin{cases} w_i(n) - \mu(l)\epsilon(l)f(n-i+1) & \text{if } (n+i)\text{mod}N = 0 \\ & \text{and } l = N \bmod(n/N) \\ w_i(n) & \text{otherwise} \end{cases} . \quad (57)$$

This algorithm updates L/N coefficients at each time step; thus, after N time instants, all coefficients have been updated using the same error signal. All other equations for this scheme are the same as that for the filtered-X LMS algorithm. For $N = 2$, this algorithm requires $1.5L + M + 1$ multiplies, $1.5L + M - 2$ adds, and $2L + M + \max(L, M+1) + 3$ memory locations per time step.

We now describe a fast exact implementation of the periodic filtered-X LMS algorithm for the choice $N = 2$. The coefficient updates over two time instants are given by

$$\mathbf{W}(n+1) = \mathbf{W}(n-1) - \mu(n-1)\epsilon(n-1)\mathbf{F}(n-1). \quad (58)$$

In addition, the correlation term $r_{xf}(n)$ has a two-time-step recursion as

$$r_{xf}(n) = r_{xf}(n-2) + x(n-1)f(n-2) - x(n-L-1)f(n-L-2). \quad (59)$$

The remaining equations are the same as those of the fast exact filtered-X LMS algorithm of Section 3.1, except that the calculations for $\widetilde{\mathbf{F}}_\Delta(n-i)$ and $\widetilde{\mathbf{F}}_\epsilon(n-i)$ are not required. Table 5 lists the equations and number of operations for this algorithm. The algorithm requires $2.5L+1.5M+4$ multiplies and $3L+1.5M+3$ adds per two time steps and uses $3L+1.5M+1.5\max(L, M+1)+3$ memory locations. With proper partitioning, $1.25L + 0.75M + 2$ multiplies per time step are required, which is a savings of $0.25(L + M)$ multiplies over the original implementation.

3.5 Sequential Filtered-X LMS Algorithm

While the periodic filtered-X LMS algorithm updates only L/N coefficients at each iteration, it uses every filtered input signal $f(n)$, such that the computational overhead associated with filtering the input signal by $\underline{\mathbf{H}}$ is not reduced. The sequential filtered-X LMS algorithm described in [19] uses every Nth value of the filtered input sequence and has similar performance to the periodic filtered-X LMS algorithm in active noise control applications. The coefficient updates for this algorithm are

$$w_i(n+1) = \begin{cases} w_i(n) - \mu(n)\epsilon(n)f(n-i+1) & \text{if } (n-i+1)\text{mod}N = 0 \\ w_i(n) & \text{otherwise} \end{cases} . \quad (60)$$

For the choice $N = 2$, the updates can be rewritten as

$$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \mathbf{W}_1(n-1) \end{bmatrix} - \begin{bmatrix} \mu(n-1)\epsilon(n-1)\widetilde{\mathbf{F}}(n-1) \\ \mu(n)\epsilon(n)\widetilde{\mathbf{F}}(n-1) \end{bmatrix}. \quad (61)$$

and $y(n)$ is computed using (1). This algorithm requires $1.5L + 0.5M + 1$ multiplies and $1.5L + 0.5M - 1$ adds at each time step and uses uses $2L + 0.5M + \max(L, M+1) + 3$ memory locations.

We now state the fast exact version of the sequential filtered-X LMS algorithm. In this case, the fast convolution technique is only applied to the input-output controller calculation, and the calculations associated with computing $f(n)$ for even time instants can be removed. Table 6 shows the operations for this version of the algorithm. It requires $2.5L+M+5$ multiplies, $3L+M+3$ adds, and $2L + 1.5M + 1.5\max(L, M+1) + 11$ memory locations to implement for two time steps. Thus, the average number of multiplies per time step is $1.25L + 0.5M + 3$, a savings of approximately $0.25L$ multiplies over the original implementation.

3.6 Summary

In summary, we have derived new implementations of several existing algorithms for feedforward active noise control. Table 7 summarizes the number of multiplies, adds, and memory locations for the standard and fast exact implementations of the algorithms. Our results indicate that the fast exact techniques are effective in reducing the complexities of these algorithms, and up to a 25% reduction in number of multiplies can be achieved in some cases.

4 Implementation Issues

Although reductions in the number of multiplies and/or adds are an indication of a new algorithm's computational efficiency, such reductions may not translate to a more-efficient real-time implementation on a chosen hardware platform. To accurately gauge such issues, one must consider the freedoms and constraints that a particular hardware platform might impose on a controller implementation, such as memory access, parallel operations, allowable operations, and load balancing. By implementing an algorithm on a chosen hardware platform, we assess an algorithm's compatibility to the hardware architecture.

In this section, we describe real-time implementations of the standard and fast exact versions of the filtered-X LMS algorithm on the Motorola DSP96002 floating-point processor running at 33MHz and sampling at 8kHz. The architecture of the Motorola DSP96002 DSP platform is similar to those of other DSPs that support parallel operations. The processor has three partitioned independently-accessible memory spaces, ten floating point data registers, and eight address registers that allow for

circular addressing and hardware-controlled loops. For a complete description of this processor, see [24, 25]. In addition, in order to allow simple complexity comparisons without radical modifications of the hand-tuned software code, we chose to fix the value of $M = L/2$ so that only one parameter (L) controls the overall complexity of the system. This choice is somewhat arbitrary but is not atypical for broadband noise control applications.

4.1 Software Implementation

All vectors calculations were implemented using circular buffers as supported by the DSP hardware. The block-of-two tilde vectors were accessed by incrementing through a given data vector array using a skip value of two. Traversing of a circular buffer by an arbitrary skip value is supported by the DSP96002 by use of modifier registers and is equivalent in complexity to traversing with a skip value of one.

One major issue that must be addressed in many real-time DSP implementations is the number and type of memory accesses required by an algorithm. This issue has not been previously considered in this paper, and it can affect the ease with which an algorithm is implemented.

In order to identify portions of the algorithms that may be memory-access-limited, we used the following simple method to estimate the effect that memory accesses might have on the efficiency of a given loop within an algorithm. Assume that memory accesses are the limiting factor in implementation speed, and ignore any data dependencies that may exist within a loop's equations. Noting that each DSP instruction takes a constant time period to complete, the time required to calculate a given equation is proportional to the number of instructions needed to implement it. So, the minimum total number of instructions required to implement any given loop is

$$\textit{Minimum Total \# of Instructions} = \frac{\textit{\# of Memory Locations Accessed}}{\textit{Max. \# of Parallel Accesses per Instruction}}. \quad (62)$$

The minimum number of instructions within a loop that is required to implement a particular equation is given by

$$\textit{Minimum \# of Instructions In Loop} = \left\lceil \frac{\textit{\# of Different Vectors}}{\textit{Max. \# of Parallel Accesses per Instruction}} \right\rceil, \quad (63)$$

where $\lceil . \rceil$ is the ceiling operation.

As an example, to calculate $y(n) = \mathbf{X}^T(n)\mathbf{W}(n)$, we need to access $2L$ quantities using a processor that allows at most 2 data memory locations to be accessed by one instruction. Thus, at least $2L/2 = L$ instructions are required to implement this operation on this processor. Since two different vectors are being accessed, we require a loop containing at least $2/2 = 1$ instruction

within it to implement this operation. Such an implementation is the most computationally-efficient one for this operation, which is to be expected given the processor design choices made by most manufacturers of DSP platforms.

Using this methodology, we determined which portions of the standard and fast exact filtered-X LMS algorithm would likely be memory-access-limited. For the standard filtered-X LMS algorithm, only the coefficient update portion of the algorithm was found to be memory-access-limited, requiring two instructions per loop iteration to perform the three memory accesses of 1) reading an element of $\mathbf{F}(n)$, 2) reading an element of $\mathbf{W}(n)$, and 3) writing an element of $\mathbf{W}(n)$. This situation is not uncommon for DSP implementations of LMS adaptive FIR filters on other processors [26] and is typically the most-efficient method for implementing a gradient-based update. For the fast exact version, the calculation of $x_w(n)$ in (7), the coefficient updates in (33), and the calculation of $\widetilde{\mathbf{F}}_\epsilon(n)$ in (36) were all found to be memory-access-limited in their original form. To simplify their implementation, we computed the vector $\mathbf{W}_\Delta(n) = \widetilde{\mathbf{W}}_0(n) + \widetilde{\mathbf{W}}_1(n)$ within the coefficient update loop and used this vector to compute $x_w(n)$ in one $L/2$-length loop. In addition, while (33) could be broken up into several different loops of length $L/2$, we chose to implement this equation in one $L/2$-length loop containing six instructions, as this minimizes the number of filter coefficient read/writes, makes the most-efficient use of the DSP96002's processor units, and preserves the complexity advantages provided by the fast exact implementation.

As mentioned previously, the fast exact algorithms in Tables 2–6 require more operations in the second time step than the first. To achieve the most-efficient implementation on a platform dedicated to the ANC task, one must balance the number of instructions required at each time step by moving computations from the more heavily-loaded time step to the less heavily-loaded time step. In addition, only portions of the algorithm that do not depend on current signal values can be moved in this way if causality is to be preserved. In our code, a portion of the coefficient updates for the fast exact filtered-X LMS algorithm at time $(n+1)$ were moved to a position in the first half of the next $(n+2)$ time step that was prior to the use of any of the elements of $\mathbf{W}(n+1)$. Such a partitioning depends on the relative values of L and M in general, which is one of the reasons why the choice $M = L/2$ was made at the onset of code development.

4.2 Experiments

Using the methodology outlined above, both the standard and fast exact versions of the filtered-X LMS algorithm were implemented on the DSP96002. Tests were then run to verify that both algorithms performed identically for a given value of L and to compare the maximum value of L

that each version would allow for the given clock and sampling rates.

First, the two implementations were tested for equivalence. Two sequences $x(n)$ and $\epsilon(n)$, each fifty samples long as shown in Figure 4, were stored within the processor for use by both algorithms. Both algorithms were then applied to this data with $\mu(n) = 0.01$ and $L = 10$. Shown in Figure 5 are the sequences $y(n)$, $f(n)$, and the first coefficient $w_0(n)$ for both the standard and fast exact versions, in which the sample-by-sample equivalence between the two algorithm's outputs can be readily seen. Here, we have only plotted the value of $w_0(n)$ for even time steps, as the fast exact version does not calculate a value for $w_0(n)$ at every time step. These results indicate that that two implementations are mathematically-equivalent.

We then ran both algorithms in a real-time setting, in which a loudspeaker generating several sinusoids of different amplitudes and frequencies between 100 and 350Hz was used as the noise-generating device. A reference signal obtained from a microphone in front of this loudspeaker was sampled and used by the processor as the input signal $x(n)$, and the output signal $y(n)$ was sent to a second loudspeaker placed approximately 1.0m in front of the noise source. A microphone placed approximately 0.5m in front of the secondary loudspeaker provided the error signal $\epsilon(n)$, and the secondary path impulse response $\underline{\mathbf{H}}$ was identified using a second LMS-based identification procedure prior to the experiment. Both the standard and fast exact versions of the filtered-X LMS algorithm were found to produce audiably-indistinguishable cancellation behavior at the error microphone for all step sizes values and filter lengths chosen.

Next, the algorithms were tested to determine the maximum value of L that both versions would allow before the processor capabilities were exceeded at the chosen 8kHz sample rate. It was found that the standard implementation enabled a controller filter length of $L_s = 672$ without failure, whereas the fast exact implementation allowed a value of $L_f = 856$, a 27.4% increase. If number of multiplies were a true indicator of algorithm complexity, we would expect a 33.3% increase given our previous results (*i.e.*, $1/0.75 = 1.333$); however, such conservative estimates ignore important factors such as 1) the overhead associated with any out-of-loop instructions and 2) the non-constant-time nature of some input/output fetching instructions.

If we assume that each instruction takes a constant time to execute, which is correct for all parallel instructions, we can count the number of instructions in the code to estimate overhead effects. The standard implementation contains $17 + 3.5L_s + k$ instructions per time step, where k is the number of instructions common to both the standard and fast exact code and L_s is the value of L for the standard implementation. The fast exact code contains a maximum of $108 + 2.625L_f + k$

instructions per time step, where L_f is the value of L for the fast exact version. Both algorithms have the maximum allowable filter lengths with equal numbers of instructions, or

$$17 + 3.5L_s + k \quad = \quad 108 + 2.625L_f + k. \tag{64}$$

After some rearrangement, we have

$$L_f \quad = \quad \frac{4}{3}L_s - \frac{104}{3} \tag{65}$$

The last term in (65) depicts the detrimental effects of overhead on the computational performance of the fast exact version. Note that if this term is ignored, we obtain $L_f = 1.333L_s$, as expected. Moreover, if we set $L_s = 672$, we find from (65) a predicted value of $L_f = 861$, close to the actual value of $L_f = 856$ observed. The difference is likely caused by the few instructions for which the constant instruction execution time assumption is not valid.

5 Conclusions

This paper describes extensions of the fast exact LMS adaptive algorithm to five different algorithms for single-channel feedforward active noise control. Savings of up to 25% in the number of multiplies can be realized with these implementations without any algebraic change to each algorithm's inherent input-output behavior. The computational savings could allow the systems to be implemented on less expensive hardware, to free up some processor cycles for other operations, or to allow the implementation of longer filter lengths on a chosen processor. Where there exist portions of the algorithms for which the block techniques are not applicable, then the savings in number of multiplies is less than 25%.

Real-time experiments with coded versions of the standard and fast exact filtered-X LMS algorithm on the Motorola DSP96002 DSP processor indicate that the latter is more efficient than the former on this processor, as the latter allows maximum filter lengths that are nearly one-third longer than those allowed for the former at the same clock and sampling rates. The implementations also illustrate the important issues of memory access and load balancing that must be considered when efficiently implementing these algorithms on a typical DSP platform. These results illustrate that the computational savings obtained by the fast exact algorithms can be obtained in a real-world setting, thus verifying their utility for actual active noise control systems. In addition, extensions of these algorithms to standard multichannel algorithms for active noise control are straightforward, and the computational benefits obtained in such cases can be expected to be similar.

Acknowledgement

This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

Biographies

David S. Nelson received the B.S. and M.S. degrees in electrical engineering from the University of Utah in 1996 and 1998, respectively. He is currently a Member of Technical Staff at L3 Communications Corp., Salt Lake City, UT. His interests include high-speed wireless modems for digital communication systems.

Scott C. Douglas received the B.S. (with distinction), M.S., and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 1988, 1989, and 1992, respectively. From 1992 to 1998, he was an Assistant Professor in the Department of Electrical Engineering at the University of Utah, Salt Lake City, UT. Since August 1998, he has been with the Department of Electrical Engineering at Southern Methodist University, Dallas, TX, as an Associate Professor. His research activities include adaptive filtering, active noise control, blind deconvolution and source separation, and VLSI/hardware implementations of digital signal processing systems.

Dr. Douglas received the Hughes Masters Fellowship Award in 1988 and the NSF Graduate Fellowship Award in 1989. He was a recipient of the NSF CAREER Award in 1995. He is the author or co-author of four book chapters and more than 80 articles in journals and conference proceedings; he also served as a section editor for *The DSP Handbook* (Boca Raton, FL: CRC/IEEE Press, 1998). He is a Senior Member of the IEEE, an associate editor for *IEEE Signal Processing Letters,* and a member of both the Neural Networks for Signal Processing Technical Committee and the Education Committee of the IEEE Signal Processing Society. He served on the Technical Program Committees of the 1995 IEEE Symposium on Circuits and Systems, Seattle, WA, and the 1998 IEEE Digital Signal Processing Workshop, Bryce Canyon, UT. He is the Proceedings Co-Chair of the 1999 Workshop on Neural Networks for Signal Processing, Madison, WI, the Technical Program Co-Chair of the 1999 DSP Workshop, Hunt, TX, and the Proceedings Editor of the 1999 International Symposium on Active Control of Sound and Vibration, Ft. Lauderdale, FL. He is on the organizing committee of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, UT. He has one patent pending. He is a frequent consultant to industry in the areas of signal processing and adaptive filtering and is a member of Phi Beta Kappa and Tau Beta Pi.

Marc Bodson received a Ph.D. degree in Electrical Engineering and Computer Science from the University of California, Berkeley, in 1986, a M.S. degree in Electrical Engineering and Computer Science and a M.S. degree in Aeronautics and Astronautics from the Massachusetts Institute of Technology, Cambridge MA, in 1982, and an Electrical and Mechanical Engineer degree from the Universite Libre de Bruxelles, Belgium, in 1980.

Currently, Marc Bodson is an Associate Professor of Electrical Engineering at the University of Utah, Salt Lake City, UT. His research interests are in adaptive control, with applications to electromechanical systems and aerospace. He is coauthor, with S. Sastry, of the book *Adaptive Control: Stability, Convergence, and Robustness* (Prentice-Hall, Englewood Cliffs, NJ, 1989). Marc Bodson has been an associate editor of the *IEEE Trans. on Control Systems Technology* since January 1996.

Marc Bodson was an Assistant Professor and an Associate Professor in the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA, between 1987 and 1993. He was a Belgian American Educational Foundation Fellow in 1980, a Visiting Lecturer at the University of California, Berkeley, in 1987, and a Lady Davis Fellow at the Technion - Israel Institute of Technology, Haifa, Israel, in 1990. He was an associate editor of the *IEEE Control Systems Magazine* from 1994 to 1995, and the chair of the technical committee on Control Electronics for the IEEE Control Systems Society from 1995 to 1997.

References

[1] G.E. Warnaka, "Active attenuation of noise–the state of the art," *Noise Contr. Eng.*, vol. 18, no. 3, pp. 100-110, May-June 1982.

[2] J.C. Stevens and K.K. Ahuja, "Recent advances in active noise control," *AIAA J.*, vol. 29, no. 7, pp. 1058-1067, July 1991.

[3] S.J. Elliott and P.A. Nelson, "Active noise control," *IEEE Signal Processing Mag.*, vol. 10, no. 4, pp.12-35, Oct. 1993.

[4] C.R. Fuller and A.H. von Flotow, "Active control of sound and vibration," *IEEE Control Sys. Mag.*, vol. 15, no. 6, pp. 9-19, Dec. 1995.

[5] E.F. Berkman and E.K. Bender, "Perspectives on active noise and vibration control," *Sound and Vibration*, vol. 31, no. 1, pp. 80-94, Jan. 1997.

[6] M.O. Tokhi and R.R. Leitch, *Active Noise Control* (Oxford: Clarendon, 1992).

[7] P.A. Nelson and S.J. Elliott, *Active Control of Sound* (London: Academic Press, 1992).

[8] C.R. Fuller, S.J. Elliott, and P.A. Nelson, *Active Control of Vibration* (London: Academic Press,1996).

[9] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (New York: Wiley, 1996).

[10] B. Widrow and E. Walach, *Adaptive Inverse Control* (Upper Saddle River, NJ: Prentice-Hall, 1996).

[11] B. Widrow, D. Shur, and S. Shaffer, "On adaptive inverse control," *Proc. 15th Asilomar Conf. Circuits, Systems, and Computers*, Pacific Grove, CA, pp. 185-195, Nov. 1981.

[12] J.C. Burgess, "Active adaptive sound control in a duct: a computer simulation," *J. Acoust. Sci. Am.*, vol. 70, pp. 715-726, 1981.

[13] E. Bjarnason, "Active noise cancellation using a modified form of the filtered–X LMS algorithm," *Proc. EUSIPCO-92, Signal Processing VI,* Brussels, Belgium, vol. 2, pp. 1053-1056, Aug. 1992.

[14] I. Kim, H. Na, K. Kim, and Y. Park, "Constraint filtered–X and filtered-U algorithms for the active control of noise in a duct," *J. Acoust. Soc. Am.*, vol. 95, no. 6, pp. 3397-3389, June 1994.

[15] M. Rupp and R. Frenzel, "Analysis of the LMS and NLMS algorithms with delayed coefficient updates under the presence of spherically invariant processes," *IEEE Trans. Signal Processing,* vol. 42, pp. 668-672, Mar. 1994.

[16] E. Bjarnason, "Analysis of the filtered–X LMS algorithm," *IEEE Trans. Speech Audio Processing,* vol. 3, pp. 504-514, Nov. 1995.

[17] Q. Shen and A.S. Spanias, "Time- and frequency-domain X–block least-mean-square algorithms for active noise control," *Noise Control Eng. J.*, vol. 44, no. 6, pp. 281-293, Nov.-Dec. 1996.

[18] M. Rupp, "Saving complexity of modified filtered–X LMS and delayed update LMS algorithms," *IEEE Trans. Circuits Syst. II: Anal. Dig. Signal Processing,* vol. 44, pp. 45-48, Jan. 1997.

[19] S.C. Douglas, "Adaptive filters employing partial updates," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Processing,* vol. 44, pp. 209-216, Mar. 1997.

[20] S.C. Douglas, "An efficient implementation of the modified filtered–X LMS algorithm," *IEEE Signal Processing Lett.,* vol. 4, pp. 286-288, Oct. 1997.

[21] M. Rupp and A.H. Sayed, "Robust FxLMS algorithms with improved convergence performance," *IEEE Trans. Speech Audio Processing,* vol. 6, pp. 78-75, Jan. 1998.

[22] Z.J. Mou and P. Duhamel, "Short-length FIR filters and their use in nonrecursive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 1322-1332, June 1991.

[23] J. Benesty and P. Duhamel, "A fast exact least mean square adaptive algorithm," *IEEE. Trans. Signal Processing*, vol. 40, pp. 2904-2912, Dec. 1992.

[24] M. El-Sharkawy, *Signal Processing, Image Processing, and Graphics Applications with Motorola's DSP96002 Processor*, vol. 1 (Englewood Cliffs, NJ: Prentice-Hall, 1994).

[25] *DSP96002: IEEE Floating-Point Dual-Port Processor User's Manual* (Motorola Inc., 1989).

[26] S. Kuo and C. Chen, "Implementation of adaptive filters with the TMS320C25 or the TMS320C30," in *Digital Signal Processing Applications with the TMS320 Family,* P. Papamichalis, ed. (Englewood Cliffs, NJ: Prentice-Hall, 1991), pp. 191-271.

[27] D.S. Nelson, S.C. Douglas, and M. Bodson, "Fast block adaptive algorithms for feedforward active noise control," *Proc. Nat. Conf. Noise Control Eng.,* Philadelphia, PA, vol. 2, pp. 209-220, June 1997.

List of Tables

List of Figures

Table 1: General notation used in this paper.

Notation	Examples	Description
Upper case letters	L and M	Constants
Lower case letters	i and n	Scalar values
Subscript values	w_i	Index of a set of scalar values
Signal(index)	$x(n)$	Value of signal at a given time step
Bold uppercase(index)	$\mathbf{X}(n)$	Vector of the last L values of a signal
Bold underlined uppercase(index)	$\underline{\mathbf{X}}(n)$	Vector of the last M values of a signal
Bold tilde uppercase(index)	$\tilde{\mathbf{X}}(n)$	Vector formed by taking every other element of a corresponding vector
subscript–Δ	$\tilde{\mathbf{X}}_{\Delta}(n)$	The sum or difference between vectors *i.e.*, $\tilde{\mathbf{X}}_{\Delta}(n) = \tilde{\mathbf{X}}(n-1) - \tilde{\mathbf{X}}(n)$
superscript–T	$\mathbf{X}^T(n)$	Vector transpose

Table 2: The fast exact filtered-X LMS algorithm.

Equation	*No. of ×'s*	*No. of +'s*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\mathbf{X}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\mathbf{F}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\epsilon_\mu(n) = \mu(n)\epsilon(n)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) - \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\mathbf{F}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2) - x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$	4	4
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\widetilde{\mathbf{F}}_\epsilon(n-1) = \widetilde{\mathbf{F}}(n-1)(\epsilon_\mu(n-1) + \epsilon_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_\epsilon(n-1) - \widetilde{\mathbf{F}}_\Delta(n)\epsilon_\mu(n) \\ \widetilde{\mathbf{F}}_\epsilon(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)\epsilon_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (two time steps)	$3L+1.5M+7$	$4L+1.5M+8$

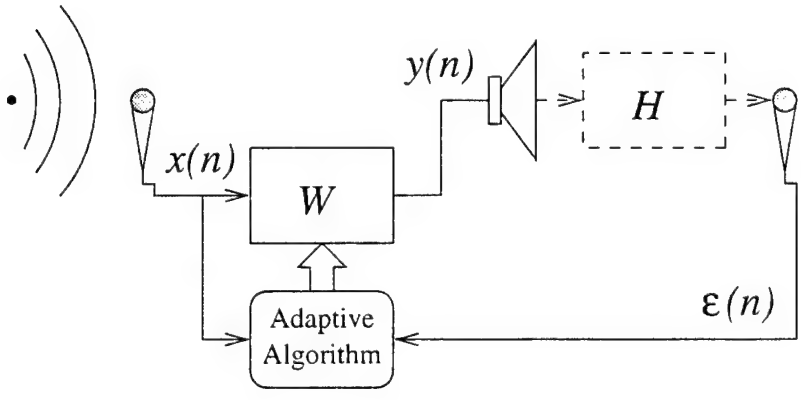

Figure 1: Block diagram of a single-channel feedforward active noise control system.

Table 3: The fast exact modified filtered-X LMS algorithm.

Equation	*No. of ×'s*	*No. of +'s*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\mathbf{X}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\mathbf{F}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$y_h(n-2) = \widetilde{\underline{\mathbf{Y}}}^T(n-2)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\underline{\mathbf{Y}}}(n-1) = \begin{bmatrix} y(n-1) \\ \widetilde{\underline{\mathbf{Y}}}(n-3) \end{bmatrix}$	0	0
$f_w(n-1) = \widetilde{\mathbf{F}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$\widetilde{\underline{\mathbf{Y}}}_\Delta(n-2) = \begin{bmatrix} y(n-3)-y(n-2) \\ \widetilde{\underline{\mathbf{Y}}}_\Delta(n-4) \end{bmatrix}$	0	1
$e_\mu(n-1) = \mu(n)(\epsilon(n-1) - (y_h(n-2) + \widetilde{\underline{\mathbf{Y}}}_\Delta^T(n-2)\widetilde{\underline{\mathbf{H}}}_1) + (f_w(n-1) + \widetilde{\mathbf{F}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)))$	$0.5L+0.5M+1$	$0.5L+0.5M+2$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\mathbf{F}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2) - x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$	4	4
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\widetilde{\underline{\mathbf{Y}}}(n) = \begin{bmatrix} y(n) \\ \widetilde{\underline{\mathbf{Y}}}(n-2) \end{bmatrix}$	0	0
$r_{ff}(n) = r_{ff}(n-2) + f(n)f(n-1) + f(n-1)f(n-2) - f(n-L)f(n-L-1) - f(n-L-1)f(n-L-2)$	4	4
$\widetilde{\underline{\mathbf{Y}}}_\Delta(n-1) = \begin{bmatrix} y(n-2)-y(n-1) \\ \widetilde{\underline{\mathbf{Y}}}_\Delta(n-3) \end{bmatrix}$	0	1
$e_\mu(n) = \mu(n)(\epsilon(n) - e_\mu(n-1)r_{ff}(n) - (y_h(n-2) - \widetilde{\underline{\mathbf{Y}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_0) + (f_w(n-1) - \widetilde{\mathbf{F}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1))$	$0.5L+0.5M+2$	$0.5L+0.5M+3$
$\widetilde{\mathbf{F}}_e(n-1) = \widetilde{\mathbf{F}}(n-1)(e_\mu(n-1) + e_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_e(n-1) - \widetilde{\mathbf{F}}_\Delta(n)e_\mu(n) \\ \widetilde{\mathbf{F}}_e(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)e_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (for two time steps)	$4.5L+3M+12$	$6L+3M+17$

Table 4: The fast exact efficient modified filtered-X LMS algorithm.

Equation	*No. of ×'s*	*No. of +'s*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n-1) = \begin{bmatrix} f(n-2)-f(n-1) \\ \widetilde{\mathbf{F}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$\underline{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \underline{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$u_h(n-1) = \widetilde{\mathbf{U}}_0^T(n-1)\widetilde{\mathbf{H}}_\Delta$	$0.5M$	$0.5M-1$
$e_\mu(n-1) = \mu(n)(\epsilon(n-2) + u_h(n-1) + (\widetilde{\mathbf{U}}_1^T(n-2) - \widetilde{\mathbf{U}}_0^T(n-2))\widetilde{\mathbf{H}}_1)$	$0.5M+1$	$M+1$
$\underline{\mathbf{R}}_{xf}(n-1) = \underline{\mathbf{R}}_{xf}(n-2) + \underline{\mathbf{X}}(n-1)f(n-1) - \underline{\mathbf{X}}(n-L-1)f(n-L-1)$	$2M$	$2M$
$\underline{\mathbf{U}}(n-1) = \begin{bmatrix} 0 \\ \underline{\mathbf{U}}(n-2) \end{bmatrix} - \underline{\mathbf{R}}_{xf}(n-1)e_\mu(n-1)$	M	M
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{F}}_\Delta(n) = \begin{bmatrix} f(n-1)-f(n) \\ \widetilde{\mathbf{F}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$\underline{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \underline{\mathbf{X}}(n-1) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) + x(n-1)f(n-2) - x(n-L)f(n-L-1) - x(n-L-1)f(n-L-2)$	4	4
$y(n) = x_w(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - e_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$e_\mu(n) = \mu(n)(\epsilon(n) - e_\mu(n-1)r_{ff}(n) + u_h(n-1) - (\widetilde{\mathbf{U}}_0^T(n-2) - \widetilde{\mathbf{U}}_{-1}^T(n+2))\widetilde{\mathbf{H}}_0$	$0.5M+2$	$0.5M+2$
$\underline{\mathbf{R}}_{fx}(n) = \underline{\mathbf{R}}_{xf}(n-1) + \underline{\mathbf{X}}(n)f(n) - \underline{\mathbf{X}}(n-L)f(n-L)$	$2M$	$2M$
$\underline{\mathbf{U}}(n) = \begin{bmatrix} 0 \\ \underline{\mathbf{U}}(n-1) \end{bmatrix} - \underline{\mathbf{R}}_{ef}(n)e_\mu(n-1)$	M	M
$r_{ff}(n) = r_{ff}(n-2) + f(n)f(n-1) + f(n-1)f(n-2) - f(n-L)f(n-L-1) - f(n-L-1)f(n-L-2)$	4	4
$\widetilde{\mathbf{F}}_e(n-1) = \widetilde{\mathbf{F}}(n-1)(e_\mu(n-1) + e_\mu(n))$	$0.5L$	1
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}_e(n-1) - \widetilde{\mathbf{F}}_\Delta(n)e_\mu(n) \\ \widetilde{\mathbf{F}}_e(n-1) + \widetilde{\mathbf{F}}_\Delta(n-1)e_\mu(n-1) \end{bmatrix}$	L	$2L$
Totals (for two time steps)	$3L+9M+12$	$4L+9.5M+14$

Table 5: The fast exact periodic filtered-X LMS algorithm.

Equation	*No. of ×'s*	*No. of +'s*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\widetilde{\underline{\mathbf{X}}}(n-1) = \begin{bmatrix} x(n-2) \\ \widetilde{\underline{\mathbf{X}}}(n-3) \end{bmatrix}$	0	0
$x_h(n-1) = \widetilde{\underline{\mathbf{X}}}^T(n-1)\widetilde{\underline{\mathbf{H}}}_\Delta$	$0.5M$	$0.5M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\mathbf{X}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n-1) = \begin{bmatrix} x(n-3)-x(n-2) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-3) \end{bmatrix}$	*previously calculated*	
$f(n-1) = x_h(n-1) + \widetilde{\underline{\mathbf{X}}}_\Delta^T(n-1)\widetilde{\underline{\mathbf{H}}}_1$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\widetilde{\underline{\mathbf{X}}}(n) = \begin{bmatrix} x(n-1) \\ \widetilde{\underline{\mathbf{X}}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$\widetilde{\underline{\mathbf{X}}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\underline{\mathbf{X}}}_\Delta(n-2) \end{bmatrix}$	*previously calculated*	
$f(n) = x_h(n-1) - \widetilde{\underline{\mathbf{X}}}_\Delta^T(n)\widetilde{\underline{\mathbf{H}}}_0$	$0.5M$	$0.5M$
$\widetilde{\mathbf{F}}(n) = \begin{bmatrix} f(n) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n) = \begin{bmatrix} x(n) \\ \widetilde{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$r_{xf}(n) = r_{xf}(n-2) + x(n-1)f(n-2) - x(n-L-1)f(n-L-2)$	2	2
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}(n-1) \\ \widetilde{\mathbf{F}}(n-2) \end{bmatrix} \epsilon_\mu(n-1)$	L	L
Totals (two time steps)	$2.5L+1.5M+4$	$3L+1.5M+3$

Table 6: The fast exact sequential filtered-X LMS algorithm.

Equation	*No. of ×'s*	*No. of +'s*
$\epsilon(n-1) = d(n-1) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-2)$	*measured*	
$\epsilon_\mu(n-1) = \mu(n-1)\epsilon(n-1)$	1	0
$\underline{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-2) \\ \underline{\mathbf{X}}(n-2) \end{bmatrix}$	0	0
$f(n-1) = \underline{\mathbf{X}}^T(n-1)\underline{\mathbf{H}}$	M	$M-1$
$\widetilde{\mathbf{X}}_\Delta(n-1) = \begin{bmatrix} x(n-2)-x(n-1) \\ \widetilde{\mathbf{X}}_\Delta(n-3) \end{bmatrix}$	0	1
$\widetilde{\mathbf{F}}(n-1) = \begin{bmatrix} f(n-1) \\ \widetilde{\mathbf{F}}(n-3) \end{bmatrix}$	0	0
$\widetilde{\mathbf{X}}(n-1) = \begin{bmatrix} x(n-1) \\ \widetilde{\mathbf{X}}(n-3) \end{bmatrix}$	0	0
$x_w(n-1) = \widetilde{\mathbf{X}}^T(n-1)(\widetilde{\mathbf{W}}_0(n-1) + \widetilde{\mathbf{W}}_1(n-1))$	$0.5L$	$L-1$
$y(n-1) = x_w(n-1) + \widetilde{\mathbf{X}}_\Delta^T(n-1)\widetilde{\mathbf{W}}_1(n-1)$	$0.5L$	$0.5L$
$\epsilon(n) = d(n) + \underline{\mathbf{H}}^T \underline{\mathbf{Y}}(n-1)$	*measured*	
$\epsilon_\mu(n) = \mu(n)\epsilon(n)$	1	0
$\widetilde{\mathbf{X}}_\Delta(n) = \begin{bmatrix} x(n-1)-x(n) \\ \widetilde{\mathbf{X}}_\Delta(n-2) \end{bmatrix}$	0	1
$r_{xf}(n) = r_{xf}(n-2) + x(n)f(n-1) - x(n-L)f(n-L-1)$	2	2
$y(n) = x_w(n-1) - \widetilde{\mathbf{X}}_\Delta^T(n)\widetilde{\mathbf{W}}_0(n-1) - \epsilon_\mu(n-1)r_{xf}(n)$	$0.5L+1$	$0.5L+1$
$\begin{bmatrix} \widetilde{\mathbf{W}}_0(n+1) \\ \widetilde{\mathbf{W}}_1(n+1) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{W}}_0(n-1) \\ \widetilde{\mathbf{W}}_1(n-1) \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{F}}(n-1)\epsilon_\mu(n-1) \\ \widetilde{\mathbf{F}}(n-1)\epsilon_\mu(n) \end{bmatrix}$	L	L
Totals (two time steps)	$2.5L+M+5$	$3L+M+3$

Table 7: Comparison of algorithm complexities.

Algorithm	*Number of $\times$'s*	*Number of $+$'s*	*Memory Locations*
FXLMS	$2L + M + 1$	$2L + M - 2$	$2L + M + \max(L, M + 1) + 3$
FE-FXLMS	$1.5L + 0.75M + 4$	$2L + 0.75M + 4$	$2.5L + 1.5M + 1.5\max(L, M + 1) + 14$
MFXLMS	$3L + 2M + 1$	$3L + 2M - 2$	$2L + 2M + \max(L, M + 1) + 3$
FE-MFXLMS	$2.25L + 1.5M + 6$	$3L + 1.5M + 9$	$2.5L + 3M + 1.5\max(L, M + 1) + 15$
EMFXLMS	$2L + 5M + 1$	$2L + 5M - 2$	$3L + 3M + 4$
FE-EMFXLMS	$1.5L + 4.5M + 6$	$2L + 4.75M + 7$	$4L + 5M + 16$
PFXLMS	$1.5L + M + 1$	$1.5L + M - 2$	$2L + M + \max(L, M + 1) + 3$
FE-PFXLMS	$1.25L + 0.75M + 3$	$1.5L + 0.75M + 4$	$3L + 1.5M + 1.5\max(L, M + 1) + 3$
SFXLMS	$1.5L + 0.5M + 1$	$1.5L + 0.5M - 1$	$2L + M + \max(L, M + 1) + 3$
FE-SFXLMS	$1.25L + 0.5M + 3$	$1.5L + 0.5M + 2$	$2L + 1.5M + 1.5\max(L, M + 1) + 11$

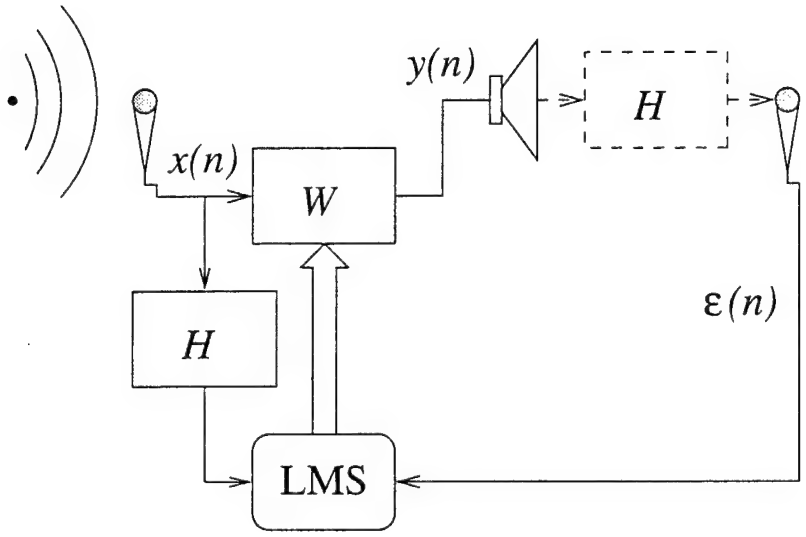

Figure 2: Block diagram of the filtered-X LMS adaptive algorithm.

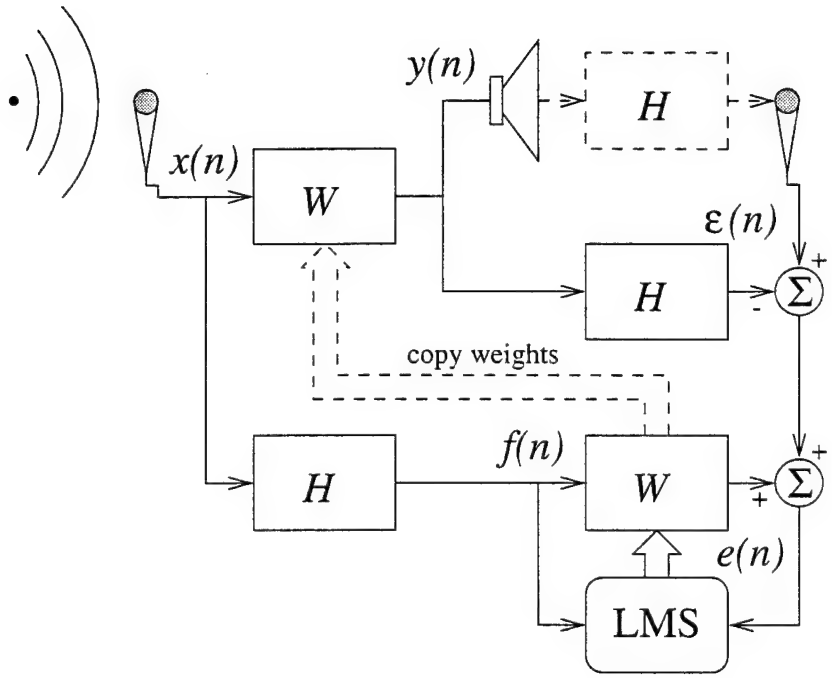

Figure 3: Block diagram of the modified filtered-X LMS adaptive algorithm.

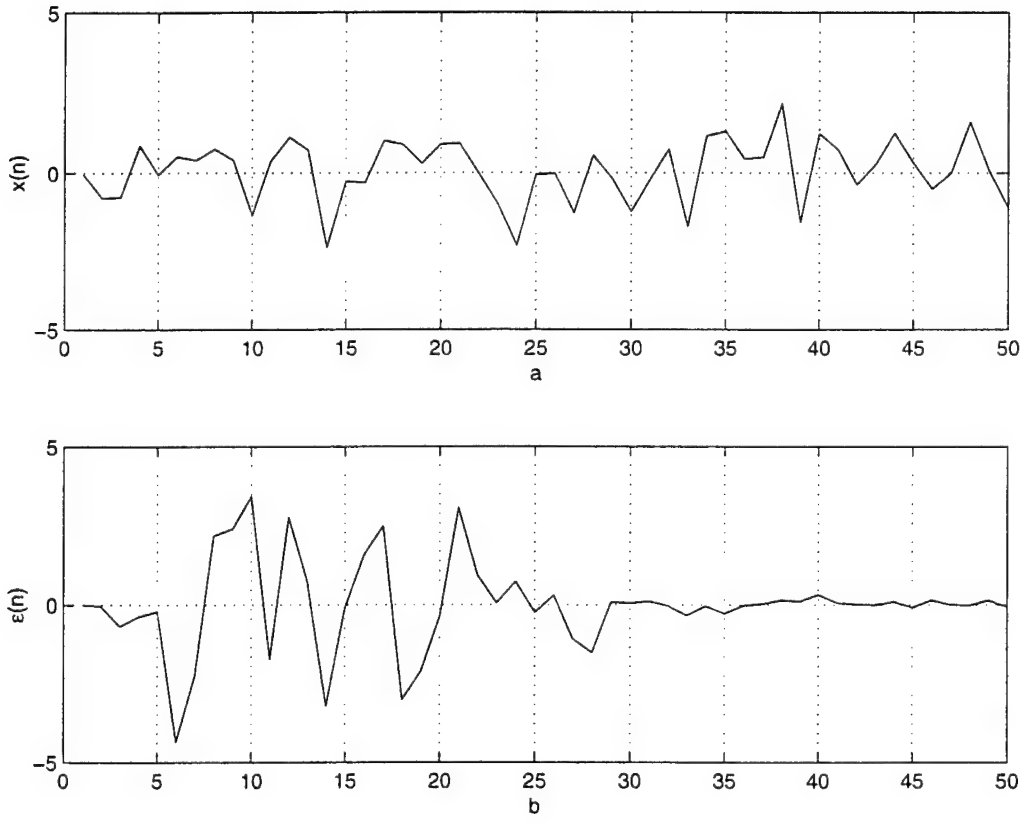

Figure 4: The input and error signals used in the first set of DSP96002 experiments: a) $x(n)$ and b) $\epsilon(n)$.

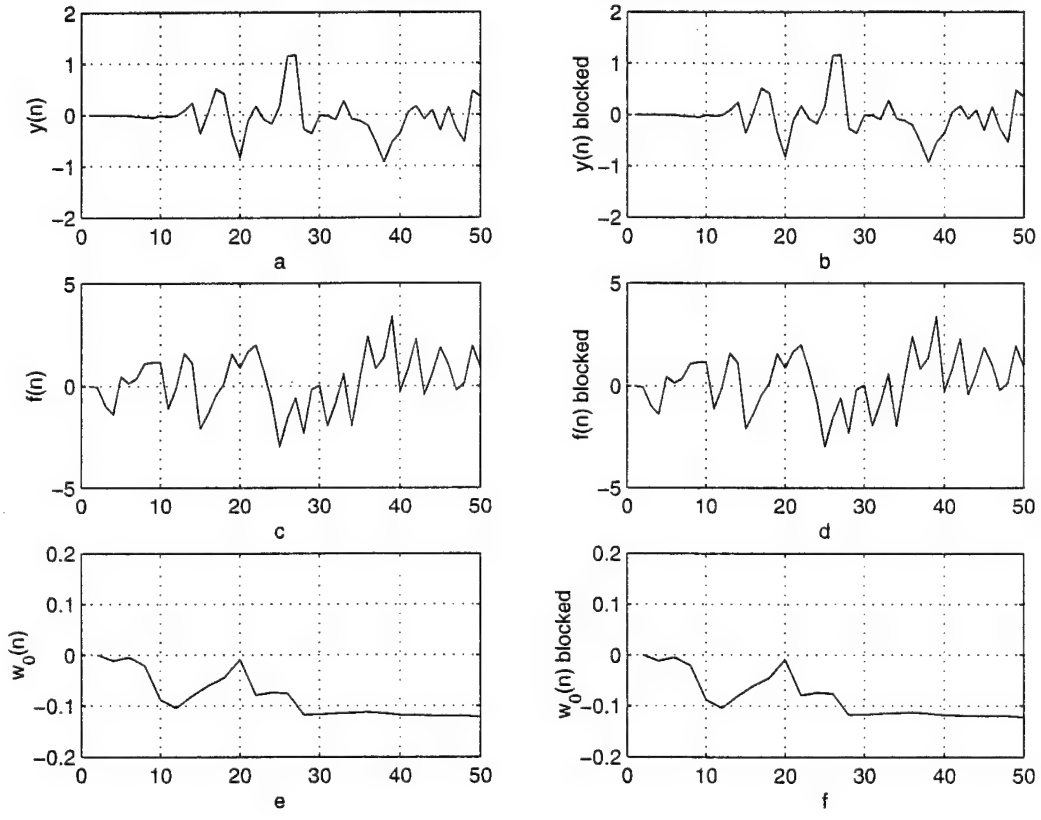

Figure 5: The signals obtained from the first set of DSP96002 experiments: a) $y(n)$ from standard version, b) $y(n)$ from fast exact version, c) $f(n)$ from standard version, d) $f(n)$ from fast exact version, e) $w_0(n)$ from standard version, and f) $w_0(n)$ from fast exact version.

A NUMERICALLY-STABLE SLIDING-WINDOW ESTIMATOR AND ITS APPLICATION TO ADAPTIVE FILTERS

S.C. Douglas and J.K. Soh
Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112 USA

ABSTRACT

While it is possible to implement the sliding discrete Fourier transform (DFT) recursively, numerical errors accumulate in the resulting outputs. This paper presents a recursive system whose output at every Lth sample time is equivalent to that of the sliding DFT and whose numerical errors do not grow with time. We provide a statistical analysis of the numerical performance of the proposed technique, showing that it outperforms a similar approximate method employing an exponentially-decaying window. When applied to sliding-window-covariance RLS adaptive filters, the technique mitigates this system's error accumulation without any significant increase in complexity. An extension of the method to the sliding discrete cosine transform (DCT) is also provided.

1. INTRODUCTION

Consider the discrete-time linear time-invariant system

$$y(k) = \sum_{m=0}^{L-1} e^{-j2\pi ml/L} x(k-m), \qquad (1)$$

where l is an integer in the range $0 \leq l \leq (L-1)$. This system computes the lth bin value of the sliding discrete Fourier transform (DFT) of the signal $x(k)$. Such a system is useful in spectral estimation [1] as well as in transform-domain adaptive filtering [2].

When $l = 0$, (1) computes the running sample average of a signal across an L-element window. This ubiquitous system is useful for numerous signal processing tasks [3]–[10]. In addition, if $x(k-m)$ represents the squared estimation error of an N-coefficient FIR adaptive filter, the recursive minimization of $y(k)$ over time yields different forms of the sliding-window-covariance recursive least-squares (SWC-RLS) adaptive filter [11, 12].

The system in (1) can be implemented recursively as

$$y(k) = e^{-j2\pi l/L} y(k-1) - x(k-L) + x(k). \qquad (2)$$

Such an implementation is quite useful, as it only requires a single (complex) multiply and two additions per time instant. However, this recursive implementation is marginally-stable, and thus finite-precision errors will linearly accumulate in the value of $y(k)$ over time. If left unchecked, such errors can make (2) useless. For this reason, the following approximate system is proposed [2]:

$$\widehat{y}(k) = \sum_{m=0}^{L-1} (\widehat{\lambda} e^{-j2\pi l/L})^m x(k-m), \qquad (3)$$

where $0 \ll \widehat{\lambda} < 1$. The recursive implementation of (3) is

$$\widehat{y}(k) = \left(\widehat{\lambda} e^{-j2\pi l/L}\right) \widehat{y}(k-1) - \widehat{\lambda}^L x(k-L) + x(k), \qquad (4)$$

which requires two multiplies and two additions per time instant. While this system is numerically-stable, it no longer computes the exact value of $y(k)$ in (1), although $|\widehat{y}(k) - y(k)|$ can be made small via suitable choice of $\widehat{\lambda}$. An alternative solution is to periodically restart the calculation of $y(k)$. This method does not require additional multiplies, but it does require more additions and memory. Moreover, the two-tiered coding strategy needed for the system's restart is undesirable for SWC-RLS adaptive filtering as it is computationally-expensive [4].

When implementing the sliding DFT, two other methods are of particular interest. The LMS spectrum analyzer uses a variant of the LMS algorithm to compute an L-point sliding DFT [13]. While its complexity is similar to that in (2), simulations in Section 4 show that it suffers from similar numerical accumulation effects when implemented in a floating-point environment, although its numerical properties in a fixed-point environment appear to be better than those of (2) [14]. When L is a power of two, the sliding FFT is a non-recursive method for computing an L-point sliding DFT that requires $O(L)$ multiplies per time instant to implement [15, 16]. This system requires $O(L \log_2 L)$ memory locations, however, which can be prohibitive for some applications.

In this paper, we present a technique that, like (4), approximates the system in (1) using a simple recursive update. Our technique is a periodically-time-varying system designed so that any numerical errors introduced by finite-precision arithmetic at time k exponentially decay to zero over time. Like (4), the new method requires two multiplications and two additions at every time instant. However, unlike (4) in which $\widehat{y}(k) \neq y(k)$ in general, our technique produces an output signal that is mathematically-equivalent to $y(k)$ in (1) at every Lth time instant. At other time instants, the difference between the output of the proposed system and $y(k)$ in (1) can be made arbitrarily small through the proper choice of the leakage parameter $\overline{\lambda}$. Moreover, the new technique does not require any

[0]This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

siginificant additional memory. An application of the new technique to SWC-RLS adaptive filters shows that it stabilizes the marginal instability of this system at almost no additional computational cost. We also provide an extension of the technique to the sliding discrete cosine transform (DCT) useful for adaptive filtering tasks [17]–[19].

2. A NUMERICALLY-STABLE RECURSIVE SLIDING-WINDOW ESTIMATOR

The proposed method for approximating (1) is a periodically-time-varying linear system. If numerical errors are not present, the system is equivalent to

$$\overline{y}(k) = \sum_{m=0}^{L-1} h_l(k,m)x(k-m) \quad (5)$$

$$h_l(k,m) = \begin{cases} e^{-j2\pi ml/L} & \text{if } k \bmod L \geq m \\ \overline{\lambda}e^{-j2\pi ml/L} & \text{if } k \bmod L < m \end{cases}, \quad (6)$$

where $x(k)$ and $\overline{y}(k)$ are the input and output signals of the system, respectively, $h_l(k,m)$ is the time-varying impulse response of the system, and $0 \ll \overline{\lambda} < 1$. When $k \bmod L = L-1$, $\overline{y}(k)$ is mathematically the same as $y(k)$ in (1). At other sample instants, $\overline{y}(k)$ deviates from $y(k)$. A simple calculation shows that

$$|\overline{y}(k) - y(k)| \leq (1-\overline{\lambda}) \sum_{m=1}^{L-1} |x(k-m)|, \quad (7)$$

such that $|\overline{y}(k) - y(k)|$ can be made arbitarily small by proper choice of $\overline{\lambda}$.

The system in (6) has a simple recursive implementation that makes it a useful for approximating the sliding DFT:

$$\overline{y}(k) = \begin{cases} \overline{\lambda}[e^{-j2\pi l/L}\overline{y}(k) - x(k-L)] + x(k), & \text{if } k \bmod L = 0 \\ e^{-j2\pi l/L}\overline{y}(k) - \overline{\lambda}x(k-L) + x(k), & \text{otherwise.} \end{cases} \quad (8)$$

This implementation requires two multiplications and two additions as well as the control logic to determine when $k \bmod L = 0$. Note that many digital signal processors (DSPs) have built-in circular pointer indexing schemes that make such a test simple to implement.

3. PERFORMANCE ANALYSIS

We now analyze the performances of the estimators in (4) and (8) in a statistical framework. For simplicity, we consider the case where $l = 0$ and $x(k)$ is real-valued, and we assume that $x(k)$ is a stationary uncorrelated random process with mean μ_x and variance σ_x^2. In each case, we determine three quantities:

- the average bias in the outputs of each system with respect to $y(k)$,
- the variance of the difference between the outputs of each system and $y(k)$, and
- the variance of the numerical error in the output of each system assuming a standard statistical model for the quantization errors.

We first analyze the performance of the system in (4) under the above assumptions. The average bias of this estimator assuming infinite precision calculations is

$$\widehat{\mu} = E\{y(k) - \widehat{y}(k)\} = E\left\{\sum_{m=0}^{L-1}(1-\widehat{\lambda}^m)x(k-m)\right\}. \quad (9)$$

Under our statistical assumptions, this expression becomes

$$\widehat{\mu} = \sum_{m=1}^{L-1}(1-\widehat{\lambda}^m)\mu_x = \left(L-1-\frac{\widehat{\lambda}-\widehat{\lambda}^L}{1-\widehat{\lambda}}\right)\mu_x. \quad (10)$$

The variance of the estimation error $y(k) - \widehat{y}(k)$, denoted as $\widehat{\sigma}^2$, is given by

$$\widehat{\sigma}^2 = E\left\{[y(k) - \widehat{y}(k) - \widehat{\mu}]^2\right\} \quad (11)$$

$$= E\left\{\left[\sum_{m=0}^{L-1}(1-\widehat{\lambda}^m)(x(k-m)-\mu_x)\right]^2\right\}. \quad (12)$$

With our assumptions, the RHS of (12) is

$$\widehat{\sigma}^2 = \left(L-1-2\frac{\widehat{\lambda}-\widehat{\lambda}^L}{1-\widehat{\lambda}} + \frac{\widehat{\lambda}^2-\widehat{\lambda}^{2L}}{1-\widehat{\lambda}^2}\right)\sigma_x^2. \quad (13)$$

We now consider the performance of (4) in a finite-precision environment, in which $\underline{\widehat{y}}(k)$, the finite-precision equivalent of $\widehat{y}(k)$, is approximately

$$\underline{\widehat{y}}(k) = \widehat{\lambda}\underline{\widehat{y}}(k-1) - \widehat{\lambda}^L x(k-L) + x(k) + \epsilon_i(k) + \epsilon_o(k) \quad (14)$$

where $\epsilon_i(k)$ and $\epsilon_o(k)$ are the numerical errors associated with the first and second terms, respectively, on the RHS of (14). Defining $\widehat{\epsilon}(k)$ as

$$\widehat{\epsilon}(k) = \underline{\widehat{y}}(k) - \widehat{y}(k), \quad (15)$$

we can develop a recursive equation for $\widehat{\epsilon}(k)$ given by

$$\widehat{\epsilon}(k) = \widehat{\lambda}\widehat{\epsilon}(k-1) + \epsilon_i(k) + \epsilon_o(k). \quad (16)$$

Assuming that $\epsilon_i(k)$ and $\epsilon_o(k)$ are zero mean and i.i.d. with variances σ_ϵ^2, we obtain the limiting variance of $\widehat{\epsilon}(k)$ as

$$\lim_{k\to\infty} \sigma_{\widehat{\epsilon}}^2(k) = \frac{2\sigma_\epsilon^2}{1-\widehat{\lambda}^2}. \quad (17)$$

We now analyze the statistical performance of (8) under our assumptions. Since the system has a time-varying impulse response, we consider quantities that are averaged over L sample times in order to provide a fair comparison. We first consider the time-averaged bias, denoted as

$$\overline{\mu} = \frac{1}{L}\sum_{n=k-L+1}^{k} E\{y(n) - \overline{y}(n)\} \quad (18)$$

$$= \frac{1}{L}\sum_{n=k-L+1}^{k} E\left\{\sum_{m=0}^{L-1}(1-h_0(n,m))x(n-m)\right\} \quad (19)$$

Employing (6), we can simply the above expression as

$$\overline{\mu} = \frac{1}{L}\sum_{m=1}^{L-1}(1-\overline{\lambda})m\mu_x = \frac{L-1}{2}(1-\overline{\lambda})\mu_x. \quad (20)$$

Table 1: Comparison of numerical performance, exponentially-windowed and new schemes, $l = 0$.

Window	*Parameter*		*Bias*		*Error Variance*		*Precision*
L	$1-\widehat{\lambda}$	$1-\overline{\lambda}$	$\widehat{\mu}/(L\mu_x)$	$\overline{\mu}/(L\mu_x)$	$\widehat{\sigma}^2/(L\sigma_x^2)$	$\overline{\sigma}^2/(L\sigma_x^2)$	$\frac{\sigma_\epsilon^2(\infty)}{(L\sigma_x^2)}$
10	1.81×10^{-3}	0.01	8.11×10^{-3}	4.50×10^{-3}	9.24×10^{-5}	4.50×10^{-5}	55.3
100	1.97×10^{-4}	0.01	9.69×10^{-3}	4.95×10^{-3}	1.26×10^{-4}	4.95×10^{-5}	50.8
$\to\infty$	$\to 0$	0.01	9.89×10^{-3}	5×10^{-3}	1.30×10^{-4}	5×10^{-5}	50.3
10	1.82×10^{-4}	0.001	8.18×10^{-4}	4.50×10^{-4}	9.40×10^{-7}	4.50×10^{-7}	550
100	1.98×10^{-5}	0.001	9.79×10^{-4}	4.95×10^{-4}	1.28×10^{-6}	4.95×10^{-7}	505
$\to\infty$	$\to 0$	0.001	9.99×10^{-4}	5×10^{-4}	1.33×10^{-6}	5×10^{-7}	500
10	1.82×10^{-5}	1×10^{-4}	8.18×10^{-5}	4.50×10^{-5}	9.42×10^{-9}	4.50×10^{-9}	5500
100	1.98×10^{-6}	1×10^{-4}	9.80×10^{-5}	4.95×10^{-5}	1.29×10^{-8}	4.95×10^{-9}	5050
$\to\infty$	$\to 0$	1×10^{-4}	1.00×10^{-5}	5×10^{-5}	1.33×10^{-8}	5×10^{-9}	5000

The time-averaged error variance of the new estimator's output, denoted as $\overline{\sigma}^2$, is defined as

$$\overline{\sigma}^2 = \frac{1}{L}\sum_{n=k-L+1}^{k} E\{[y(n)-\overline{y}(n)-\overline{\mu}]^2\} \quad (21)$$

$$= \frac{1}{L}\sum_{n=k-L+1}^{k} E\left\{\left[\sum_{m=0}^{L-1}(1-h_0(n,m))(x(n-m)-\mu_x)\right]^2\right\} \quad (22)$$

After some simplification, the RHS of (22) becomes

$$\overline{\sigma}^2 = \frac{1}{L}\sum_{m=1}^{L-1}(1-\overline{\lambda})^2 m\sigma_x^2 = \frac{L-1}{2}(1-\overline{\lambda})^2\sigma_x^2. \quad (23)$$

Finally, we consider the accumulation of errors in $\overline{y}(k)$ in a finite-precision environment. The numerical model in this case is similar to that in (14) and is given by

$$\underline{\overline{y}}(k) = \begin{cases} \overline{\lambda}[\underline{\overline{y}}(k)-x(k-L)]+x(k)+\epsilon_o(k)+\epsilon_i(k), \\ \qquad \text{if } k \bmod L = 0 \\ \underline{\overline{y}}(k)-\overline{\lambda}x(k-L)+x(k)+\epsilon_i(k), \\ \qquad \text{otherwise.} \end{cases} \quad (24)$$

For simplicity, we only consider the worst-case performance by determining the level of numerical errors in $\underline{\overline{y}}(k)$ at time instants $k = nL-1$. By defining $\overline{\epsilon}(nL-1) = \underline{\overline{y}}(nL-1) - y(nL-1)$, it can be shown that

$$\overline{\epsilon}(nL-1) = \overline{\lambda}\overline{\epsilon}((n-1)L-1)+\epsilon_o((n-1)L) + \sum_{m=1}^{L}\epsilon_i(nL-m). \quad (25)$$

Thus, the limiting value of the variance of $\overline{\epsilon}(nL-1)$ is

$$\lim_{n\to\infty}\sigma_{\overline{\epsilon}}^2(nL-1) = \frac{(L+1)\sigma_\epsilon^2}{1-\overline{\lambda}^2}. \quad (26)$$

We now compare the performances of the recursive estimators in (4) and (8). For this comparison, we equate the numerical precision error variances as expressed by (17) and (26), respectively, and solve for $\widehat{\lambda}$ in terms of $\overline{\lambda}$:

$$\widehat{\lambda} = \sqrt{1-\frac{2}{L+1}(1-\overline{\lambda}^2)}. \quad (27)$$

Then, we compute the average biases and error variances for each algorithm using their respective $\overline{\lambda}$ and $\widehat{\lambda}$. Table 1 shows the values of the biases $\widehat{\mu}$ and $\overline{\mu}$ as well as the error variances $\widehat{\sigma}^2$ and $\overline{\sigma}^2$, normalized by $L\mu_x$ and $L\sigma_x^2$, respectively. In each case, it is observed that the proposed scheme has less bias and less error variance than the exponentially-windowed scheme. Thus, the proposed scheme is to be preferred from the standpoint of numerical performance.

4. SIMULATIONS

4.1. Sliding DFT

We now explore the numerical performance of the sliding DFT in (2), the proposed sliding DFT in (8), and the LMS spectrum analyzer in [13], via simulations in the MATLAB floating-point computational environment. In each case, we have computed the $L = 16$-point sliding DFT of a sequence of unit-variance Gaussian random variables using each method, and we then calculate the total squared numerical error between the resulting 16 signals and those obtained from MATLAB's `fft()` function at each time instant. For the proposed method, we have chosen $\overline{\lambda} = 0.999$.

Fig. 1 shows the total squared numerical errors for the three methods as a function of time, where we have only plotted the errors when $k \bmod L = L-1$ such that all of the methods would compute the exact DFT in an infinite-precision environment. As can be seen, the numerical errors of both the standard sliding DFT and LMS spectrum analyzer grow linearly over time, whereas that of the proposed technique does not exhibit such growth. While the growth in numerical errors are not severe in this floating-point compuational environment, these results indicate that the proposed technique mitigates numerical growth of errors, unlike the other recursive methods.

4.2. Recursive Least-Squares Adaptive Filtering

In sliding-window-covariance RLS adaptive filters, one minimizes the error criterion

$$\mathcal{J}(\mathbf{W}(k)) = \sum_{m=0}^{L-1}[d(k-m)-\mathbf{W}^T(k)\mathbf{X}(k-m)]^2, \quad (28)$$

where $d(k)$ is a desired response signal, $\mathbf{X}(k) = [x(k) \cdots x(k-N+1)]^T$ is the input signal vector, and $\mathbf{W}(k) = [w_0(n) \cdots w_{N-1}(k)]^T$ is the FIR filter coefficient vector at time k. The cost function minimum occurs at

$$\mathbf{W}(k) = \mathbf{R}^{-1}(k)\mathbf{P}(k), \quad (29)$$

where

$$\mathbf{R}(k) = \sum_{m=0}^{L-1}\mathbf{X}(k-m)\mathbf{X}^T(k-m) \quad (30)$$

$$\mathbf{P}(k) = \sum_{m=0}^{L-1} d(k-m)\mathbf{X}(k-m). \quad (31)$$

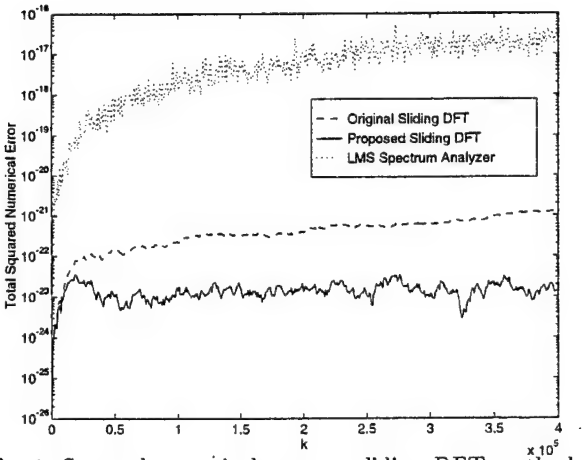

Fig. 1: Squared numerical errors – sliding DFT methods.

Comparing $\mathbf{R}(k)$ and $\mathbf{P}(k)$ with $y(k)$ in (1), it is clear that we can recursively compute them in a similar manner. Such a calculation is marginally-stable, however, and therefore, any implementation that explicitly or implicitly employs such recursions experiences numerical error accumulation in a finite-precision environment.

Given our previous results, it is natural to consider a modified cost function of the form

$$\overline{\mathcal{J}}(\overline{\mathbf{W}}(k)) = \sum_{m=0}^{L-1} h_0(k,m)[d(k-m) - \overline{\mathbf{W}}^T(k)\mathbf{X}(k-m)]^2 \quad (32)$$

where $h_0(k,m)$ is as defined in (6). A similar derivation to that which produces (29)–(31) can be performed for $\overline{\mathcal{J}}(\overline{\mathbf{W}}(k))$, and the solution obtained is identical to (29) except that the corresponding $\overline{\mathbf{R}}(k)$ and $\overline{\mathbf{P}}(k)$ are identical in form to $\overline{y}(k)$ in (5). Thus, we compute $\overline{\mathbf{R}}(k)$ recursively as

$$\overline{\mathbf{R}}(k) = \begin{cases} \lambda[\overline{\mathbf{R}}(k) - \mathbf{X}(k-L)\mathbf{X}^T(k-L)] + \mathbf{X}(k)\mathbf{X}^T(k) \\ \qquad \text{if } k \bmod L = 0 \\ \overline{\mathbf{R}}(k) - \lambda\mathbf{X}(k-L)\mathbf{X}^T(k-L) + \mathbf{X}(k)\mathbf{X}^T(k) \\ \qquad \text{otherwise,} \end{cases} \quad (33)$$

and $\overline{\mathbf{P}}(k)$ has a similar update. The resulting recursive solution for $\overline{\mathbf{W}}(k)$ is numerically-stable, because the updates for $\overline{\mathbf{R}}(k)$ and $\overline{\mathbf{P}}(k)$ are numerically-stable. Moreover, because $\overline{\mathcal{J}}(\overline{\mathbf{W}}(k)) = \mathcal{J}(\mathbf{W}(k))$ for $k = nL-1$, we have $\overline{\mathbf{W}}(nL-1) = \mathbf{W}(nL-1)$ when numerical errors are absent.

By applying the matrix inversion lemma [20] twice to the RHS of (33) and combining the resulting update with that for $\overline{\mathbf{P}}(k)$, one obtains a set of direct updates for $\overline{\mathbf{W}}(k)$. Table 2 lists the equations for this numerically-stable version of the $O(N^2)$ SWC-RLS algorithm, where $\overline{\mathbf{W}}(k) = \overline{\mathbf{W}}_L(k)$. This algorithm's complexity is identical to that of the original SWC-RLS algorithm ($\lambda = 1$) when $k \bmod L \neq 0$. Thus, the two algorithm's complexities are similar when the block length L is large relative to the filter length N.

We now provide simulations comparing the numerical performance of the algorithm in Table 2 with respect to the original $O(N^2)$ SWC-RLS algorithm in the MATLAB floating-point computational environment. In these simulations, $x(k)$ is a zero-mean uncorrelated Gaussian signal with unit variance, and $d(k)$ is generated from $\mathbf{X}(k)$ as

$$d(k) = \mathbf{X}^T(k)\mathbf{W}_{opt} + \eta(k), \quad (34)$$

Table 2: The $O(N^2)$ stabilized SWC-FTF algorithm.

if $k \bmod L = 0$

$$\mathbf{C}(k) = \frac{\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}{\overline{\lambda} + \mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}$$

$$\mathbf{R}_{L+1}^{-1}(k) = \overline{\lambda}^{-1}\left(\mathbf{R}_L^{-1}(k-1) - \mathbf{C}(k)\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\right)$$

else

$$\mathbf{C}(k) = \frac{\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}{1 + \mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)\mathbf{X}(k)}$$

$$\mathbf{R}_{L+1}^{-1}(k) = \mathbf{R}_L^{-1}(k-1) - \mathbf{C}(k)\mathbf{X}^T(k)\mathbf{R}_L^{-1}(k-1)$$

end

$$\epsilon(k) = d(k) - \mathbf{X}^T(k)\overline{\mathbf{W}}_L(k-1)$$

$$\overline{\mathbf{W}}_{L+1}(k) = \overline{\mathbf{W}}_L(k-1) + \epsilon(k)\mathbf{C}(k)$$

$$\mathbf{D}(k) = \mathbf{R}_{L+1}^{-1}(k)\mathbf{X}(k-L)$$

$$\gamma(k) = \left(-\overline{\lambda}^{-1} + \mathbf{X}^T(k-L)\mathbf{D}(k)\right)^{-1}$$

$$\mathbf{R}_L^{-1}(k) = \mathbf{R}_{L+1}^{-1}(k) - \mathbf{D}(k)\mathbf{D}^T(k)\gamma(k)$$

$$\nu(k) = d(k-L) - \mathbf{X}^T(k-L)\overline{\mathbf{W}}_{L+1}(k)$$

$$\overline{\mathbf{W}}_L(k) = \overline{\mathbf{W}}_{L+1}(k) + \gamma(k)\nu(k)\mathbf{D}(k)$$

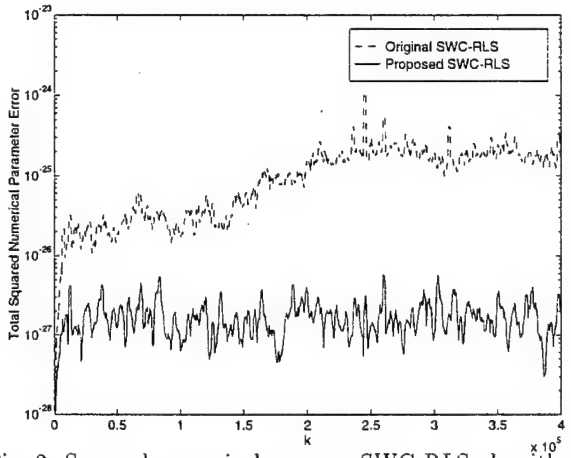

Fig. 2: Squared numerical errors – SWC-RLS algorithms.

where $\mathbf{W}_{opt} = [1\ 1\ \cdots\ 1]^T$, $N = 10$, $L = 20$, and $\eta(k)$ is a zero-mean uncorrelated Gaussian signal with variance $\sigma_\eta^2 = 0.0001$. In this case, we compute the quantity $||\overline{\mathbf{W}}(k) - \mathbf{W}_{opt}(k)||^2$ with $\overline{\lambda} = 0.995$, where $\mathbf{W}_{opt}(k)$ is obtained by directly minimizing the cost function in (28) at each iteration. Fig. 2 shows a smoothed version of this quantity for all $k \bmod L = L-1$. As can be seen, the numerical errors remain small for the proposed implementation. For comparison, we also show the quantity $||\mathbf{W}(k) - \mathbf{W}_{opt}(k)||^2$ for the standard SWC-RLS algorithm, in which the linear growth of the numerical errors is readily apparent. These results indicate the ability of the new method to stabilize the marginally-stable SWC-RLS updates. Such results are clearly useful in real-time situations, particularly when fixed-point computations are employed.

5. EXTENSIONS

When processing real-valued signals, a real-valued transform such as the discrete cosine transform (DCT) is more-useful than the DFT for certain applications. A recursive method for computing the lth bin value of the sliding DCT

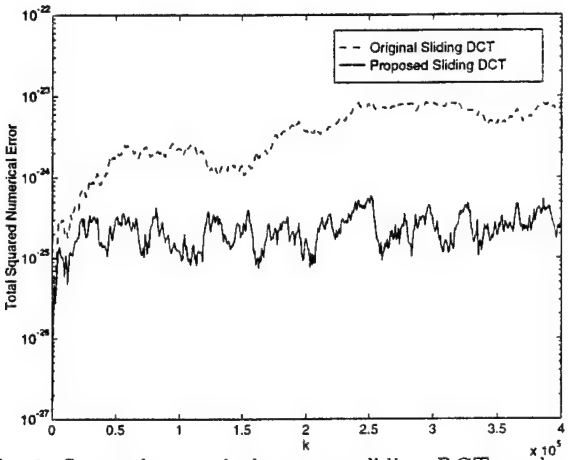

Fig. 3: Squared numerical errors – sliding DCT methods.

for $1 \leq l \leq L-1$ is [17]

$$u_l(k) = \gamma_l y(k) \tag{35}$$

$$\begin{aligned} y(k) &= 2\cos(l\pi/L)y(k-1) - y(k-2) + x(k) - x(k-1) \\ &\quad - (-1)^l[x(k-L) - x(k-L-1)], \end{aligned} \tag{36}$$

where $\gamma_l = \sqrt{2/L}\cos(l\pi/(2L))$ for $1 \leq l \leq L-1$ and $\gamma_0 = 1/\sqrt{L}$.

Similar to (8), we can define a periodically-time-varying system that implements the sliding DCT in a numerically-stable fashion. This system replaces $y(k)$ in (35) by

$$\overline{y}(k) = \begin{cases} \overline{\lambda}[2\cos(l\pi/L)y(k-1) - y(k-2)] + x(k) \\ - x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \\ \qquad \text{if } k \bmod L = 0 \\ 2\cos(l\pi/L)y(k-1) - \overline{\lambda}y(k-2) + x(k) \\ - x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \\ \qquad \text{if } k \bmod L = 1 \\ 2\cos(l\pi/L)y(k-1) - y(k-2) + x(k) \\ - x(k-1) - \overline{\lambda}(-1)^l[x(k-L) - x(k-L-1)] \\ \qquad \text{otherwise.} \end{cases} \tag{37}$$

In this case, a second-order recursive system is employed, and $\overline{\lambda}$ is applied to the recursive terms within the updates in different ways. We have tested the numerical performance of this implementation in a manner similar to that used to test the sliding DCT techniques previously. As the results in Fig. 3 show, the proposed technique mitigates the growth in numerical errors experienced by (35)–(36).

As the proposed methods produce a mathematically-accurate result for $k \bmod L = L-1$, we can easily extend the interval over which $\overline{y}(k) = y(k)$ for each system. For the DFT and DCT methods in (8) and (37), respectively, we replace the test $k \bmod L = 0$ with the test $k \bmod M = 0$, where $M > L$. Then, for $L \leq k \bmod M < M$, we use the marginally-stable updates (2) and (36) to compute the outputs of each system, respectively. The value of $\overline{y}(k)$ for these systems is mathematically-identical to the sliding DFT and DCT over the intervals $L-1 \leq k \bmod M < M$. A similar approach can be applied to the SWC-RLS algorithm in Table 2. While potentially-useful, this extension reduces the average number of stabilizing updates by the factor L/M over that of the original method, and thus one can expect that the levels of numerical error in the outputs will increase by a factor of approximately M/L.

6. CONCLUSIONS

In this paper, we have presented recursive systems that implement the sliding DFT and DCT in a numerically-stable fashion. These algorithms employ periodically-time-varying linear systems that mitigate the growth in numerical errors when implemented in finite-precision arithmetic while exactly computing the desired quantities over a portion of the input sequence. Moreover, we have applied this technique to the $O(N^2)$ sliding-window-covariance RLS adaptive filter, showing that the linear accumulation of numerical errors of the original system does not occur in the proposed system. These methods are expected to have wide use in a number of practical signal processing systems, including the implementation of adaptive filtering algorithms.

REFERENCES

[1] S. Kay, *Modern Spectral Estimation: Theory and Application* (Englewood Cliffs, NJ: Prentice-Hall, 1988).

[2] J.J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Mag.*, vol. 9, no. 1, pp. 14-37, Jan. 1992.

[3] J. Benesty and P. Duhamel, "A fast exact least mean square adaptive algorithm," *IEEE. Trans. Signal Processing*, vol. 40, pp. 2904-2912, Dec. 1992.

[4] S. Gay and S. Tavathia, "The fast affine projection algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 5, pp. 3023-3026, May 1995.

[5] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 2, pp. 945-948, May 1995.

[6] S.C. Douglas, "The fast affine projection algorithm for active noise control," *Proc. 29th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, vol. 2, pp. 1245-1249, Oct. 1995.

[7] S.C. Douglas, "Fast exact filtered-X LMS and LMS algorithms for multichannel active noise control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Munich, Germany, vol. 1, pp. 399-402, Apr. 1997.

[8] Q. Zhu, S.C. Douglas, and K.F. Smith, "A pipelined architecture for LMS adaptive filters without adaptation delay," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Munich, Germay, vol. 3, pp. 1933-1936, Apr. 1997.

[9] D.S. Nelson, S.C. Douglas, and M. Bodson, "Fast block adaptive algorithms for feedforward active noise control," *Proc. Nat. Conf. Noise Control Eng.*, State College, PA, vol. 2, pp. 197-208, June 1997.

[10] S.C. Douglas, "An efficient implementation of the modified filtered-X LMS algorithm," *IEEE Signal Processing Lett.*, vol. 4, pp. 286-288, Oct. 1997.

[11] J.M. Cioffi and T. Kailath, "Windowed fast transversal filter adaptive algorithms with normalization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 607-625, June 1985.

[12] D.T.M. Slock, "Underdetermined growing and sliding window covariance fast transversal filter RLS algorithms," *Proc. Signal Processing VI (EUSIPCO)*, Brussels, Belgium, vol. 2, pp. 1169-1172, Aug. 1992.

[13] B. Widrow, P. Baudrenghien, M. Vetterli, and P.F. Titchener, "Fundamental relations between the LMS algorithm and the DFT," *IEEE Trans. Circ. Syst.*, vol. 34, pp. 814-819, July 1987.

[14] F. Beaufays and B. Widrow, "On the advantages of the LMS spectrum analyzer over nonadaptive implementations of the sliding-DFT," *IEEE Trans. Circuits Systems II: Fund. Theory App.*, vol. 42, pp. 218-220, Apr. 1995.

[15] M.M. Covell, "An algorithm design environment for signal processing," Ph.D. thesis, Research Laboratory of Electronics, MIT, Cambridge, MA, 1989.

[16] B. Farhang-Boroujeny, "$O(N)$-complexity transform domain adaptive filters," *IEEE Trans. Circ. Syst. II: Anal. Dig. Signal Processing*, vol. 42, pp. 478-480, July 1995.

[17] S.S. Narayan, A.M. Peterson, and M.J. Narasimha, "Transform domain LMS algorithm," *IEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 609-619, June 1983.

[18] D.F. Marshall, W.K. Jenkins, and J.J. Murphy, "The use of orthogonal transforms for improving the performance of adaptive filters," *IEEE Trans. Circ. Syst.*, vol. 36, pp. 474-484, Apr. 1989.

[19] F. Beaufays, "Transform-domain adaptive filters: an analytical approach," *IEEE Trans. Signal Processing*, vol. 43, pp. 422-431, Feb. 1995.

[20] T. Kailath, *Linear Systems Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1980).

DELAY COMPENSATION METHODS FOR STOCHASTIC GRADIENT ADAPTIVE FILTERS

S.C. Douglas and J.K. Soh
Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112 USA

ABSTRACT

The coefficient delays within the updates of the delayed LMS and filtered–X LMS algorithms can limit the performances of these schemes in practical applications. Previously, delay compensation methods have been proposed to overcome these difficulties. In this paper, we present a unifying view of these delay compensation methods and derive a new efficient implementation of the modified filtered–X LMS algorithm. We also consider and present efficient delay-compensated algorithms for systems in which the delay path is time-varying. Simulations verify the useful performances of the schemes.

1. INTRODUCTION

Stochastic gradient adaptive filters are useful for applications in communications, control, and signal processing. In some applications, however, it can be difficult to implement even the simplest structures. This difficulty is caused by the dependence of the filter error at time n on the most-recent coefficient values. In VLSI implementations of the least-mean-square (LMS) adaptive filter, for example, this dependence makes the pipelining of the system a challenge. Approximate versions of the adaptive filter that introduce delay within the coefficient updates, such as the delayed LMS algorithm [1, 2], are often employed. In feedforward adaptive control systems, the filtered–X LMS algorithm employs a distributed form of delay to approximate a gradient descent procedure on a mean-squared error surface [3, 4].

Although useful, these modified algorithms usually perform worse than their original counterparts due to the delays incurred within the algorithms' updates [1, 4]. Recently, various techniques have been developed that compensate for the feedback delays within the delayed LMS and filtered–X LMS algorithms [5]–[11]. These techniques achieve performance that is similar to that of the LMS algorithm while enabling the practical implementation of these systems. Little is known about the structure and relationships between these methods, however, nor is it clear whether additional implementations are possible. Moreover, none of these methods apply to systems with time-varying delay paths, such as the filtered–X LMS algorithm with an on-line plant impulse response estimator [3].

In this paper, we provide a unifying view of the delay compensation methods presented in [5]–[11] by studying their general algebraic structure, and a new implementation of the modified filtered–X LMS algorithm is derived. In addition, we consider systems with a time-varying feedback delay and provide an efficient delay-compensated algorithm for this case. Simulations verify the usefulness of the methods for feedforward control tasks.

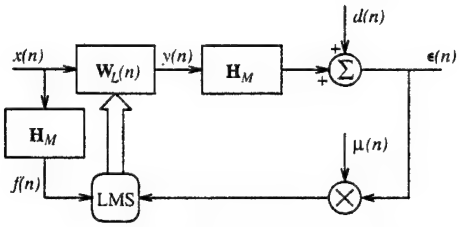

Fig. 1: The filtered–X LMS algorithm.

2. DELAY COMPENSATION FOR ADAPTIVE FEEDFORWARD CONTROL

2.1. Filtered–X LMS Algorithm

For convenience, we first review the filtered–X LMS algorithm, which includes the delayed LMS algorithm as a special case. Fig. 1 shows the structure of this algorithm, in which an output actuator signal $y(n)$ is computed from an input sensor signal $x(n)$ as

$$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l), \qquad (1)$$

where $w_l(n)$, $0 \leq l \leq L-1$ are the L controller coefficients at time n. The output signal propagates to the error sensor, where it combines with the desired response signal $d(n)$ to produce the error signal

$$\epsilon(n) = d(n) + \sum_{m=1}^{M} h_m y(n-m), \qquad (2)$$

where h_m, $1 \leq m \leq M$ is the plant impulse response.

In the filtered–X LMS algorithm, we update $w_l(n)$ as

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)f(n-l), \qquad (3)$$

where the filtered input signal is computed as

$$f(n) = \sum_{m=1}^{M} h_m x(n-m). \qquad (4)$$

This algorithm requires $2L+M+1$ multiply/accumulates (MACs) per iteration to implement, assuming that $\epsilon(n)$ is a measured quantity. To obtain the delayed LMS algorithm, set $h_m = -\delta_{m-M}$, where M is the amount of delay.

[0]This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

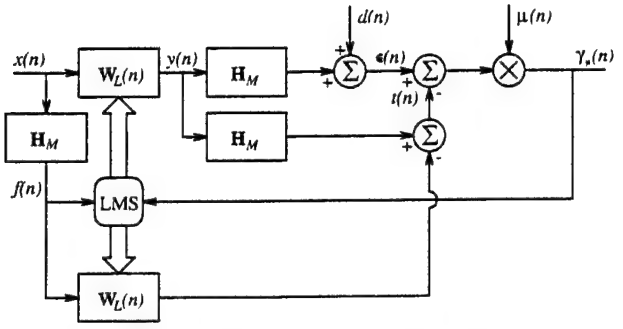

Fig. 2: The modified filtered–X LMS algorithm.

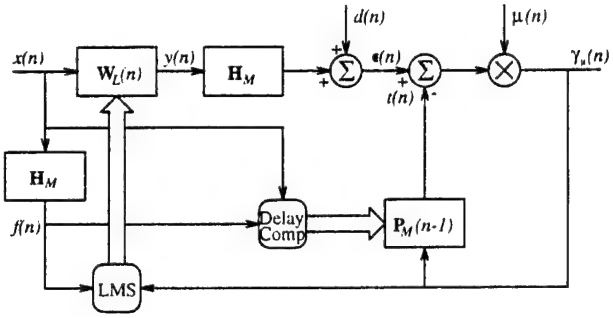

Fig. 3: Alternate modified filtered–X LMS algorithm.

2.2. The Modified Filtered–X LMS Algorithm

Because the error $\epsilon(n)$ contains delayed coefficients $\{w_l(n-i)\}$, $1 \le i \le M$, the filtered–X LMS algorithm's adaptation performance is degraded with respect to an equivalent LMS algorithm that uses the error

$$\gamma(n) \;=\; d(n) + \sum_{l=0}^{L-1} w_l(n) f(n-l). \qquad (5)$$

In [5], a method for calculating $\gamma(n)$ from the error sensor signal $\epsilon(n)$ is given. This algorithm has been termed the modified filtered–X LMS algorithm [5], the constraint filtered–X LMS algorithm [6], and the LMS algorithm for active noise control [9]. This technique computes the correction term

$$t(n) \;=\; \sum_{m=1}^{M} h_m y(n-m) - \sum_{l=0}^{L-1} w_l(n) f(n-l), \qquad (6)$$

such that

$$\gamma(n) \;=\; \epsilon(n) - t(n). \qquad (7)$$

Then, the coefficients are updated as

$$w_l(n+1) \;=\; w_l(n) - \gamma_\mu(n) f(n-l), \qquad (8)$$

where $\gamma_\mu(n) = \mu(n)\gamma(n)$. Fig. 2 shows the block diagram of this system. If the impulse response h_m used to compute $t(n)$ accurately models the characteristics of the physical plant, then this algorithm's behavior is similar to that of an LMS algorithm with desired response signal $d(n)$ and input signal $f(n)$.

Examining this algorithm, it is found that it requires $3L + 2M + 1$ MACs, which is $L + M$ more MACs than the filtered–X LMS algorithm. If L is large, then this algorithm's complexity can prohibit its real-time implementation.

2.3. Alternate Implementations of the Modified Filtered–X LMS Algorithm

In [8]–[11], alternative forms of the modified filtered–X LMS algorithm are developed. In this section, we explore the general structure of these delay-compensation methods and develop a new implementation. For this development, we first express the correction term in (6) as

$$t(n) \;=\; \sum_{i=1}^{M} h_i \left\{ y(n-i) - \sum_{j=0}^{L-1} x(n-i-j) w_j(n) \right\} \qquad (9)$$

$$\;=\; \sum_{i=1}^{M} h_i \sum_{j=0}^{L-1} x(n-i-j)\{w_j(n-i) - w_j(n)\} \qquad (10)$$

By iterating (8) backwards in time by i iterations, we obtain

$$w_j(n) - w_j(n-i) \;=\; -\sum_{k=1}^{i} \gamma_\mu(n-k) f(n-j-k) \qquad (11)$$

Substituting (11) into the RHS of (10), we obtain

$$t(n) \;=\; \sum_{i=1}^{M} h_i \sum_{j=0}^{L-1} x(n-i-j) \times \sum_{k=1}^{i} \gamma_\mu(n-k) \sum_{l=1}^{M} h_l x(n-j-k-l), \qquad (12)$$

where we have used (4) to expand $f(n-j-k)$ in (11).

The four summations on the RHS of (12) can be ordered in $4! = 24$ different ways, and these orderings can produce different algorithms. It appears, however, that only a limited number of these algorithms are both computationally-efficient and useful. Two of these cases are now considered.

Case 1: By ordering the the summations in (12) in terms of their indices as $\{i, k, j, l\}$, we have

$$t(n) = \sum_{i=1}^{M} h_i \sum_{k=1}^{i} \gamma_\mu(n-k) \sum_{j=0}^{L-1} x(n-i-j) f(n-k-j) \qquad (13)$$

Define

$$r_{xf,m}(n) \;=\; \sum_{j=0}^{L-1} x(n-j-m) f(n-j). \qquad (14)$$

Then, we can express (13) as

$$t(n) \;=\; \sum_{i=1}^{M} h_i \sum_{k=1}^{i} r_{xf,i-k}(n-k) \gamma_\mu(n-k). \qquad (15)$$

Directly computing $t(n)$ as in (15) requires $O(M^2)$ operations [8]. However, by defining

$$u_i(n) \;=\; \sum_{k=0}^{i-1} r_{xf,i-k}(n-k) \gamma_\mu(n-k), \qquad (16)$$

it can be easily shown that

$$u_i(n) \;=\; \begin{cases} r_{xf,0}(n)\gamma_\mu(n) & \text{if } i = 1 \\ u_{i-1}(n-1) + r_{xf,i}(n)\gamma_\mu(n) & \text{if } 2 \le i \le M \end{cases} \qquad (17)$$

Table 1: Alternate modified filtered–X LMS algorithm.

Equation	*MACs*
$\gamma_\mu(n) = \mu(n)\left\{\epsilon(n) - \sum_{k=1}^{M} p_k(n-1)\gamma_\mu(n-k)\right\}$	$M+1$
$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l)$	L
$f^{(M+1)}(n) = f^{(M+1)}(n-L) = 0$	0
for $j = M$ to 2 step (-1) do	
$\quad f^{(j)}(n) = f^{(j+1)}(n) + h_j x(n-j+1)$	$M-1$
$\quad f^{(j)}(n-L) = f^{(j+1)}(n-L) + h_j x(n-j-L+1)$	$M-1$
$\quad p_j(n) = p_j(n-1) + f(n-j+1)f^{(j)}(n) - f(n-j-L+1)f^{(j)}(n-L)$	$2M-2$
end	
$f(n+1) = f^{(2)}(n) + h_1 x(n)$	1
$p_1(n) = p_1(n-1) + f(n+1)f(n) - f(n-L+1)f(n-L)$	2
for $l = 0$ to $L-1$ do	
$\quad w_l(n+1) = w_l(n) - \gamma_\mu(n)f(n-l)$	L
end	
Total: $2L+5M$	

Such a computation is of reasonable complexity because $r_{xf,i}(n)$ can be easily updated as

$$r_{xf,i}(n) = r_{xf,i}(n-1) + x(n-i)f(n) - x(n-L-i)f(n-L). \quad (18)$$

The resulting algorithm is identical to that in [11]. This implementation requires $2L+5M+1$ MACs to implement, which is simpler than that in [8] when $M > 4$.

Case 2: We now consider the ordering of the summations in (12) as $\{k, j, l, i\}$. Then, the expression becomes

$$t(n) = \sum_{k=1}^{M} \gamma_\mu(n-k) \sum_{j=0}^{L-1} f(n-j-k) \sum_{i=k}^{M} h_i x(n-i-j) \quad (19)$$

Define

$$p_k(n) = \sum_{j=0}^{L-1} f(n-j-k+1) \sum_{i=k}^{M} h_i x(n-i-j+1). \quad (20)$$

Then, we have

$$t(n) = \sum_{k=1}^{M} p_k(n-1)\gamma_\mu(n-k). \quad (21)$$

Furthermore, by defining

$$f^{(k)}(n) = \sum_{i=k}^{M} h_i x(n-i+1), \quad (22)$$

then both $p_k(n)$ and $f^{(k)}(n)$ have simple combined order- and time-recursive updates, given by

$$f^{(k)}(n) = \begin{cases} h_M x(n-M+1) & \text{if } k = M \\ f^{(k+1)}(n) + h_k x(n-k+1) & \text{if } k < M \end{cases} \quad (23)$$

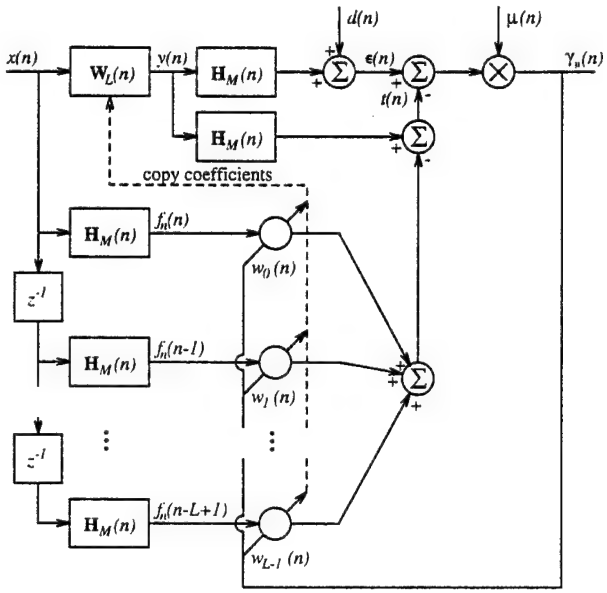

Fig. 4: Modified filtered–X LMS algorithm for time-varying plants.

$$p_k(n) = p_k(n-1) + f(n-k+1)f^{(k)}(n) - f(n-k-L+1)f^{(k)}(n-L). \quad (24)$$

Table 1 lists the operations of this new algorithm. It requires $2L+5M$ MACs to implement, a complexity that is almost identical to that of the algorithm in Case 1 above. Fig. 3 shows the structure of this algorithm, in which the delay compensation appears as a time-varying IIR filter acting on the error signal $\epsilon(n)$.

3. DELAY COMPENSATION FOR TIME-VARYING PLANTS

When the plant impulse response $h_m(n)$ varies with time, the filtered–X LMS algorithm suffers an additional performance degradation. Define

$$f_i(n) = \sum_{m=1}^{M} h_m(i)x(n-m). \quad (25)$$

The filtered–X LMS algorithm uses $f(n-l) = f_{n-l}(n-l)$ to update $w_l(n)$ in (3). If an instantaneous known change occurs in the plant impulse response, the value of $f(n-l)$ is incorrect for l time instants following the change. When L is large, this difference can cause an extended period of divergence in the controller coefficients. This problem is not solved by any of the methods in Section 2.

To provide correct convergence behavior for the filtered–X LMS algorithm, one should compute $f_n(n-l)$, $0 \le l \le L-1$ at each time instant and use $f_n(n-l)$ in place of $f(n-l)$ in (3). If delay compensation for time-varying plants is also desired, then $t(n)$ is calculated as

$$t(n) = \sum_{m=1}^{M} h_m(n)y(n-m) - \sum_{l=0}^{L-1} w_l(n)f_n(n-l), \quad (26)$$

and $\gamma(n)$ is used in place of $\epsilon(n)$ within the updates. Fig. 4 shows the structure of this modified filtered–X LMS algorithm for time-varying plants. This algorithm requires

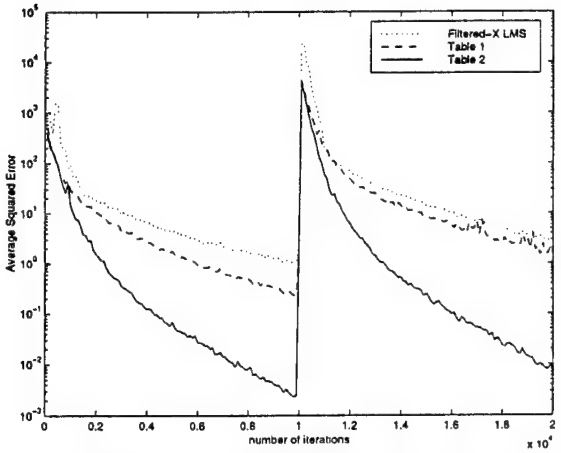

Fig. 5: Average squared errors of the competing schemes.

$LM + 3L + M + 1$ MACs to implement and is significantly more complex than the algorithms presented previously.

To simplify the algorithm's implementation, we first relate the filtered–X LMS algorithm to the delay compensation portion of the fast affine projection algorithm [12, 13]. It can be shown that the alternate implementation of the filtered–X LMS algorithm described in Section III of [10], when used with a time-varying plant, implements the filtered–X LMS algorithm with $f(n-l)$ replaced by $f_n(n-l)$. Then, by combining this algorithm with a slightly-modified version of the delay compensation method in [11], we obtain an efficient implementation of the modified filtered–X LMS algorithm for time-varying plants [14]. Table 2 lists the equations for this algorithm, which requires $2L + M^2 + 6M + 1$ MACs at each iteration. For $L > M(M+5)/(M+1)$, this version is more efficient than that in Fig. 4, and its performance is much improved over the algorithms of Section 2 when $h_m(n)$ is time-varying.

4. SIMULATIONS

We now verify the performances of the algorithms via simulations. Fig. 5 shows the average squared error $E\{\epsilon^2(n)\}$ for the filtered–X LMS ($\mu(n) = 0.0004$), the alternative modified filtered–X LMS ($\mu(n) = 0.0006$), and the efficient modified filtered–X LMS ($\mu(n) = 0.0016$) algorithms for time varying plants, respectively, as averaged from 20 simulation runs. In this case, $d(n)$ was generated by applying a ten-tap unity-coefficient filter to a white Gaussian signal $x(n)$ with $E\{x^2(n)\} = 1$, $h_m = 0.9^m$, $M = 10$, and $L = 100$. At time $n = 10000$, the values of h_m were inverted, and the chosen values of $\mu(n)$ provided the fastest convergence behavior without divergence in each case. The algorithm in Table 2 performs the best of the three, and the algorithm in Table 1 outperforms the filtered–X LMS algorithm.

5. CONCLUSIONS

We have considered methods for compensating the feedback delays of the delayed LMS and filtered–X LMS algorithms. From the structure of the calculations, we have derived a new implementation of the modified filtered–X LMS algorithm. We also have presented a novel adaptation algorithm for systems with time-varying plants. Simulations indicate that the delay-compensated algorithms can outperform the filtered–X LMS algorithm. The proposed methods are expected to be useful for a number of tasks in communications, control, and signal processing.

Table 2: Efficient modified filtered–X LMS algorithm for time-varying plants.

Equation	*MACs*
for $m = 0$ to M do	
$r_m(n) = r_m(n-1) + x(n)x(n-m) - x(n-L)x(n-L-m)$	$2M+2$
end	
$\gamma_\mu(n) = \mu(n)\left\{\epsilon(n) - \sum_{m=1}^{M} h_m(n)u_m(n-1)\right\}$	$M+1$
$y(n) = \sum_{l=0}^{L-1} \widehat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n)$	$L+M-1$
$c_1(n) = h_1(n)\gamma_\mu(n)$	1
$e_1(n) = c_1(n)$	
for $m = 1$ to $M-1$ do	
$c_{m+1}(n) = h_{m+1}(n)\gamma_\mu(n)$	$M-1$
$e_{m+1}(n) = e_m(n-1) + c_{m+1}(n)$	$M-1$
end	
$u_1(n) = \sum_{q=1}^{M} c_q(n)r_q(n)$	M
for $m = 1$ to $M-1$ do	
$u_{m+1}(n) = u_m(n-1) + \sum_{q=1}^{M} c_q(n)r_{m-q}(n-q)$	$M(M-1)$
end	
for $l = 0$ to $L-1$ do	
$\widehat{w}_l(n+1) = \widehat{w}_l(n) - e_M(n)x(n-M-l)$	L
end	
Total: $2L + M^2 + 6M + 1$	

REFERENCES

[1] G. Long, F. Ling, and J.G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1397-1405, Sept. 1989.

[2] N.R. Shanbhag and K.K. Parhi, *Pipelined Adaptive Digital Filters* (Boston, MA: Kluwer, 1994).

[3] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (New York: Wiley, 1996).

[4] E. Bjarnason, "Analysis of the filtered–X LMS algorithm," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 504-514, Nov. 1995.

[5] E. Bjarnason, "Active noise cancellation using a modified form of the filtered–X LMS algorithm," *Proc. 6th European Sig. Proc. Conf.* vol. 2 (Amsterdam: Elsevier, 1992), pp. 1053-1056.

[6] I. Kim, H. Na, K. Kim, and Y. Park, "Constraint filtered–X and filtered–U algorithms for the active control of noise in a duct," *J. Acoust. Soc. Am.*, vol. 95, no. 6, pp. 3397-3389, June 1994.

[7] R.D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," *IEEE Signal Processing Lett.*, vol. 2, p. 223, Dec. 1995.

[8] M. Rupp, "Saving complexity of modified filtered–X LMS and delayed update LMS algorithms," *IEEE Trans. Circuits Syst. II: Anal. Dig. Signal Processing*, vol. 44, pp. 45-48, Jan. 1997.

[9] S.C. Douglas, "Fast exact filtered–X LMS and LMS algorithms for multichannel active noise control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Munich, Germany, vol. 1, pp. 399-402, Apr. 1997.

[10] S.C. Douglas, "Reducing the computational and memory requirements of the multichannel filtered–X LMS adaptive controller," *Proc. Nat. Conf. Noise Contr. Eng.*, University Park, PA, vol. 2, pp. 209-220, June 1997.

[11] S.C. Douglas, "An efficient implementation of the modified filtered-X LMS algorithm," *IEEE Signal Processing Lett.*, vol. 4, pp. 286-288, Oct. 1997.

[12] S. Gay and S. Tavathia, "The fast affine projection algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 5, pp. 3023-3026, May 1995.

[13] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 2, pp. 945-948, May 1995.

[14] S.C. Douglas, "Method and apparatus for multichannel active noise and vibration control," patent pending.

Fast Implementations of the Filtered-X LMS and LMS Algorithms for Multichannel Active Noise Control

Scott C. Douglas, *Senior Member, IEEE*

***Abstract*—In some situations where active noise control could be used, the well-known multichannel version of the filtered–X least mean square (LMS) adaptive filter is too computationally complex to implement. In this paper, we develop a fast, exact implementation of this adaptive filter for which the system's complexity scales according to the number of filter coefficients within the system. In addition, we extend computationally efficient methods for effectively removing the delays of the secondary paths within the coefficient updates to the multichannel case, thus yielding fast implementations of the LMS adaptive algorithm for multichannel active noise control. Examples illustrate both the equivalence of the algorithms to their original counterparts and the computational gains provided by the new algorithms.**

***Index Terms*— Acoustic noise, active noise control, adaptive control, adaptive filters, adaptive signal processing, least mean square methods, vibration control.**

I. Introduction

INTEREST in active methods for the suppression of noise and vibration has grown recently, as evidenced by the numerous review articles and books that have appeared on the subject [1]–[9]. Although the potential for active noise and vibration control has long been recognized [10], successful implementations of these techniques have begun to appear only recently. Such success can be attributed to the rapid maturation of technology in three areas: 1) novel electroacoustic transducers, 2) advanced adaptive control algorithms, and 3) inexpensive and reliable digital signal processing (DSP) hardware. As advances in these areas are developed, active suppression of noise and vibration can be expected to find wider use in a number of commercial, industrial, and military applications. In this paper, we focus on the algorithms used in multichannel active noise and vibration control systems as implemented in DSP hardware.

Perhaps the most popular adaptive control algorithm used in DSP implementations of active noise and vibration control systems is the filtered–X least-mean-square (LMS) algorithm [11]. There are several reasons for this algorithm's popularity. First, it is well-suited to both broadband and narrowband control tasks, with a structure that can be adjusted according to the problem at hand. Second, it is easily described and understood, especially given the vast background literature on adaptive filters upon which the algorithm is based [12], [13]. Third, its structure and operation are ideally suited to the architectures of standard DSP chips, due to the algorithm's extensive use of the multiply/accumulate (MAC) operation. Fourth, it behaves robustly in the presence of physical modeling errors and numerical effects caused by finite-precision calculations. Finally, it is relatively simple to set up and tune in a real-world environment.

Despite its popularity, the standard filtered–X LMS algorithm suffers from one drawback that makes it difficult to implement when a multichannel controller is desired: the complexity of the coefficient updates for the finite impulse response (FIR) filters within the controller in these situations is much greater than the complexity of the input–output calculations. It is not unusual for the coefficient updates of the standard implementation to require more than *ten times* the number of MAC's needed to compute the outputs of the controller for fixed coefficient values, and the situation worsens as the number of error sensors is increased. For this reason, recent efforts have focused on ways to reduce the complexity of the filtered–X LMS algorithm in a multichannel context. Suggested changes include: 1) block processing of the coefficient updates using fast convolution techniques [14], 2) partial updating of the controller coefficients [15], and 3) filtered-error methods [16]–[18]. While useful, these methods often reduce the overall convergence performance of the controller, either because they introduce additional delays into the coefficient update loop or because they throw away useful information about the state of the control system. Such a performance loss may not be tolerable in some applications.

In addition to these computational difficulties, the multichannel filtered–X LMS algorithm also suffers from excessive data storage requirements. This algorithm employs filtered input signal values that are created by filtering every input signal by every output-actuator-to-error-sensor channel of the acoustic plant. The number of these terms can be an order-of-magnitude greater than the number of controller coefficients and input signal values used in the input–output calculations. As typical DSP chips have limited on-chip memory, system designers may be forced to use costly off-chip memory within their controller architectures that can further slow the operation of the system due to limits in input/output data throughput.

Manuscript received June 5, 1996; revised Aguust 19, 1998. This material was based on work supported in part by the U.S. Army Research Office under Contract DAAH04-96-1-0085 and in part by the National Science Foundation under Grant MIP-9501680. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dennis R. Morgan.

The author is with the Department of Electrical Engineering, School of Engineering and Applied Science, Southern Methodist University, Dallas, TX 75275 USA (e-mail: douglas@seas.smu.edu).

Publisher Item Identifier S 1063-6676(99)04628-3.

While some of the aforementioned techniques for complexity reduction also have reduced memory requirements, the performance of the overall system is effectively limited by these methods.

A third limitation of the multichannel filtered–X LMS adaptive controller is due to the propagation delays caused by the physical distances between the output actuators and the error sensors. Because of these delays, the error signals contain delayed versions of the controller coefficients, and these delays lead to a reduced stability range for the stepsize parameter and slower convergence speeds [19]. If the impulse responses of the secondary paths between the output actuators and the error sensors can be accurately estimated, then it is possible to approximately calculate the true LMS adaptive updates for the controller filters, as described in [20] and [21] in the single-channel case. However, a straightforward extension of this idea to the multichannel case yields an algorithm with approximately twice the complexity of the original filtered–X LMS controller. More recently, techniques for efficiently calculating the LMS adaptive updates for a single-channel controller have been provided in [22]–[24]. These techniques have not been extended to the multichannel case, however, and any additional simplifications resulting from such an extension have not been explored.

In this paper, we present novel methods for reducing the computational and memory requirements of the multichannel filtered–X LMS and multichannel LMS adaptive controllers. Our solutions are alternative implementations of these systems that are mathematically equivalent to the original implementations, and thus they preserve the characteristic robust and accurate behaviors of the algorithms. The complexity and memory requirements of the new implementations, however, are significantly reduced over those of the original implementations, especially for controllers with a large number of channels. Moreover, since the filtered-input signals are not needed in our implementations, the excessive memory requirements of the original implementations are avoided.

This paper is organized as follows. For simplicity of discussion, Section II presents the original as well as our novel implementation of the filtered–X LMS algorithm in the single-channel case, although the new implementation's computational savings are only realized in the multichannel case. The multichannel extensions are provided in Section III, along with illustrative examples indicating the computational savings obtained with the new implementation. In Section IV, we provide two extensions of the method of calculating the LMS coefficient updates for an adaptive controller in [23] to the multichannel case, showing how the algorithm can be integrated with the efficient multichannel algorithm in Section III. Example simulations in Section V show the equivalence of the new algorithms to their more complex counterparts, and simple methods for mitigating the marginal stabilities of the sliding-window calculations within the new algorithms are provided. As for mathematical notation, scalar variables are employed throughout the paper to enable the algorithms' direct translation to DSP processor code, and indices of parameter sets are for the most part lower-case versions of the variable designating the number of parameters; e.g., $w_l(n)$ for $0 \leq l \leq L-1$.

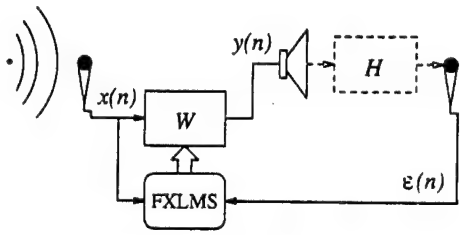

Fig. 1. Single-channel filtered–X LMS adaptive controller.

II. Single-Channel Filtered–X LMS Algorithms

A. Standard Implementation

To simplify our discussion, we initially present the single-channel filtered–X LMS adaptive feedforward controller; the multichannel filtered–X LMS algorithm is described in Section III. Fig. 1 shows a block diagram of this system, in which a sensor placed near a sound source collects samples of the input signal $x(n)$ for processing by the system. This system computes an actuator output signal $y(n)$ using a time-varying FIR filter of the form

$$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l) \tag{1}$$

where $w_l(n)$, $0 \leq l \leq L-1$ are the controller coefficients at time n and L is the controller filter length. The acoustic output signal produced by the controller combines with the sound as it propagates to the quiet region, where an error sensor collects the combined signal. We model this error as

$$\epsilon(n) = d(n) + \sum_{m=-\infty}^{\infty} h_m y(n-m) \tag{2}$$

where $d(n)$ is the undesired sound as measured at the error sensor and h_m, $-\infty < m < \infty$ is the plant impulse response. Note that (2) is never computed as $\epsilon(n)$ is a measurement of a physical quantity. In addition, (2) assumes that the secondary propagation path is linear and time-invariant. Although changes in room acoustics can occur over time and loudspeakers often have nonlinear transfer characteristics at low frequencies and high driving levels, we assume for simplicity throughout this paper that (2) is an accurate model for the error sensor signal.

The filtered–X LMS coefficient updates are given by

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)f(n-l) \tag{3}$$

where $\mu(n)$ is the algorithm step size at time n, the filtered input sequence $f(n)$ is computed as

$$f(n) = \sum_{m=1}^{M} h_m x(n-m) \tag{4}$$

and M is the FIR filter length of an appropriate estimate of the plant impulse response. In practice, the values of h_m used in (4) are estimates of the actual h_m in (2) and are usually obtained in a separate estimation procedure that is performed

prior to the application of control. For notational simplicity, we will not distinguish the differences in these two parameter sets in what follows. A discussion of the performance effects caused by errors in the estimates of h_m can be found in [25].

A study of (1), (3), and (4) shows that the filtered–X LMS algorithm requires $2L + M + 1$ MAC operations and $2L + \max\{L, M + 1\} + 1$ memory locations to store the necessary $w_l(n)$, h_m, $x(n-l)$, and $f(n-l)$ for the algorithm at each step. For typical choices of the controller and plant filter lengths, the complexity and memory requirements of this algorithm are reasonable. As will be shown, however, such is not the case for the natural extension of this algorithm to the multichannel control task.

B. New Implementation

We now describe a new implementation of the single-channel filtered–X LMS algorithm [26]–[28]. This method combines the adjoint LMS/corrected phase filtered error (CPFE) algorithm [17], [18] with a method for delay compensation used in fast projection adaptive filters [29], [30]. To derive the implementation, we write the coefficient updates of the original algorithm in the form

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n) \sum_{m=1}^{M} h_m x(n-l-m). \quad (5)$$

Define $\varepsilon_m(n)$ for $1 \le m \le M$ as

$$\varepsilon_m(n) = h_m \mu(n)\epsilon(n). \quad (6)$$

Then, (5) becomes

$$w_l(n+1) = w_l(n) - \sum_{m=1}^{M} \varepsilon_m(n)x(n-l-m). \quad (7)$$

We can represent the relation in (7) for M successive time steps as

$$w_l(n+1) = w_l(n-M+1) - \sum_{p=0}^{M-1} \sum_{m=1}^{M} \varepsilon_m(n-p) \cdot x(n-l-m-p). \quad (8)$$

We can expand the summation on the right-hand-side of (8) in a particularly useful way as in (9), shown at the bottom of the page, where we define the lth auxiliary coefficient $\hat{w}_l(n)$ as

$$\hat{w}_l(n) = w_l(n-M+1) - \sum_{p=1}^{M-1} \sum_{m=p+1}^{M} \varepsilon_m(n-M+m-p) \cdot x(n-l-M-p). \quad (10)$$

The expression in (9) indicates an important fact about the structure of the filtered–X LMS updates: the same input sample $x(n-l-m)$ is used in successive time instants to update the same coefficient $w_l(n)$. We can exploit this structure to develop a set of coefficient updates that are grouped according to the individual $x(n-l-m)$ values appearing on the right-hand-side of (9). Such a scheme updates the lth auxiliary coefficient $\hat{w}_l(n)$ rather than the actual controller coefficient $w_l(n)$. Define $e_m(n)$ as

$$e_m(n) = \sum_{p=1}^{m} \varepsilon_p(n-m+p). \quad (11)$$

Then, it is straightforward to show that $\hat{w}_l(n)$ can be updated as

$$\hat{w}_l(n+1) = \hat{w}_l(n) - e_M(n)x(n-M-l). \quad (12)$$

Thus, $\hat{w}_l(n+1)$ is obtained by subtracting from $\hat{w}_l(n)$ the last column of terms on the RHS of (9).

Since $e_M(n)$ is obtained by filtering $\mu(n)\epsilon(n)$ by the time-reversed plant impulse response $\{h_M, h_{M-1}, \cdots, h_1\}$, (12) is the single-channel version of the adjoint LMS/CPFE algorithm [17], [18]. What is novel is the relationship in (9) that provides the link between $\hat{w}_l(n)$ and $w_l(n)$, or, equivalently, the link between the adjoint LMS/CPFE and filtered–X LMS algorithms. We can use (9) to compute $y(n)$ for the filtered–X LMS algorithm using $\hat{w}_l(n)$ as calculated by (12). To proceed, we substitute the expression for $w_l(n+1)$ in (7) into (9). Using (11), we obtain

$$w_l(n) = \hat{w}_l(n) - \sum_{m=1}^{M-1} e_m(n-1)x(n-l-m-1). \quad (13)$$

Substituting the expression for $w_l(n)$ in (13) into (1), we produce the equivalent expression

$$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{l=0}^{L-1} \sum_{m=1}^{M-1} e_m(n-1) \cdot x(n-l-m-1)x(n-l). \quad (14)$$

Define the correlation term $r_m(n)$ as

$$r_m(n) = \sum_{l=0}^{L-1} x(n-l-m)x(n-l). \quad (15)$$

Then, (14) becomes

$$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n). \quad (16)$$

$$\begin{aligned} w_l(n+1) = \hat{w}_l(n) \\ - \varepsilon_1(n)x(n-l-1) - \varepsilon_2(n)x(n-l-2) \cdots - \varepsilon_M(n)x(n-l-M) \\ - \varepsilon_1(n-1)x(n-l-2) \cdots - \varepsilon_{M-1}(n-1)x(n-l-M) \\ \ddots \quad \vdots \\ - \varepsilon_1(n-M+1)x(n-l-M) \end{aligned} \quad (9)$$

Such a calculation is of reasonable complexity because $r_m(n)$ can be recursively updated as

$$r_m(n) = r_m(n-1) + x(n)x(n-m) - x(n-L)x(n-L-m). \quad (17)$$

Moreover, $e_m(n)$ has a simple order-recursive update of the form

$$e_m(n) = \begin{cases} h_1\epsilon_\mu(n), & \text{if } m = 1 \\ e_{m-1}(n-1) + h_m\epsilon_\mu(n), & \text{if } 2 \le m \le M \end{cases} \quad (18)$$

where

$$\epsilon_\mu(n) = \mu(n)\epsilon(n). \quad (19)$$

Collecting (12) and (16)–(19), we obtain a set of equations that exactly computes the output signal of the filtered–X LMS adaptive controller. This algorithm requires $2L + 4M - 1$ MAC's to implement at each iteration. Thus, this version is more computationallycomplex than the original implementation of the filtered–X LMS algorithm, which only requires $2L + M + 1$ MAC's per iteration. In the multichannel case, however, the alternative implementation can save operations and memory storage, as we now show.

III. Multichannel Filtered–X LMS Algorithms

A. Standard Implementation

We now describe the multichannel version of the filtered–X LMS algorithm in its original implementation [7], [8]. In multichannel control, I input sensors are used to collect I input signals $x^{(i)}(n)$, $1 \le i \le I$. The controller computes J output signals $y^{(j)}(n)$, $1 \le j \le J$ as

$$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} w_l^{(i,j)}(n)x^{(i)}(n-l) \quad (20)$$

where $w_l^{(i,j)}(n)$, $0 \le l \le L-1$ are the L FIR filter coefficients for the ith-input-to-jth-output channel of the controller. The J controller output signals propagate to the desired quiet region, where K error sensors measure the error signals $\epsilon^{(k)}(n)$, $1 \le k \le K$ as

$$\epsilon^{(k)}(n) = d^{(k)}(n) + \sum_{j=1}^{J} \sum_{m=-\infty}^{\infty} h_m^{(j,k)} y^{(j)}(n-m) \quad (21)$$

and $h_m^{(j,k)}$, $-\infty < m < \infty$ is the jth-output-to-kth-error plant impulse response channel.

In the original filtered–X LMS algorithm, IJK filtered input signals $f^{(i,j,k)}(n)$ are computed as

$$f^{(i,j,k)}(n) = \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-m) \quad (22)$$

from which $w_l^{(i,j)}(n)$ is updated as

$$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) f^{(i,j,k)}(n-l) \quad (23)$$

where

$$\epsilon_\mu^{(k)}(n) = \mu(n)\epsilon^{(k)}(n). \quad (24)$$

A careful study of the filtered–X LMS algorithm described by (20) and (22)–(24) reveals the fact that this implementation requires $IJK(L+M)+K$ MAC's to compute the coefficient updates, even though computing the controller outputs only requires IJL MAC's. Thus, the complexity of the update calculations is more than K times the complexity of the input–output calculations. For systems with a large number of error sensors, the computational burden of the coefficient updates can overwhelm the capabilities of the processor chosen for the control task.

The standard implementation of the filtered–X LMS algorithm also has memory requirements that can exceed the capabilities of a chosen processor. The total storage needed is $IJ(K+1)L + JKM + I\max(L, M+1) + K$, and for long controller filter lengths, the bulk of this storage is for the $IJKL$ filtered input signals $f^{(i,j,k)}(n-l)$. Clearly, it is desirable to find alternative implementations of the filtered–X LMS algorithm that have reduced computational and memory requirements. We now present an algorithm that is based on the method described in Section II.

B. New Implementation

We consider the multichannel extension of the new version of the filtered–X LMS algorithm in Section II-B. To determine the appropriate grouping of terms for the updates, we substitute the expression for $f^{(i,j,k)}(n-l)$ in (22) into the update in (23) to get

$$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-l-m) \quad (25)$$

$$= w_l^{(i,j)}(n) - \sum_{m=1}^{M} \varepsilon_m^{(j)}(n) x^{(i)}(n-l-m) \quad (26)$$

where we have defined the JM terms $\varepsilon_m^{(j)}(n)$ for $1 \le j \le J$ and $1 \le m \le M$ as

$$\varepsilon_m^{(j)}(n) = \sum_{k=1}^{K} h_m^{(j,k)} \epsilon_\mu^{(k)}(n). \quad (27)$$

Because (26) and (7) are similar in form, we can use a method analogous to that in Section II-B to implement the multichannel filtered–X LMS algorithm. Define

$$e_m^{(j)}(n) = \sum_{p=1}^{m} \varepsilon_p^{(j)}(n-m+p). \quad (28)$$

Then, we can define a set of IJL auxiliary coefficients $\hat{w}_l^{(i,j)}(n)$ whose updates are given by

$$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l). \quad (29)$$

TABLE I
FAST MULTICHANNEL FILTERED–X LMS ALGORITHM

Equation	*MACs*
for $m = 2$ to M do	
$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}$	$2I(M-1)$
end	
for $k = 1$ to K do	
$\epsilon_\mu^{(k)}(n) = \mu(n)\epsilon^{(k)}(n)$	K
end	
for $j = 1$ to J do	
$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n)$	$IJL + J(M-1)$
$e_1^{(j)}(n) = \sum_{k=1}^{K} h_1^{(j,k)}\epsilon_\mu^{(k)}(n)$	JK
for $m = 1$ to $M-1$ do	
$e_{m+1}^{(j)}(n) = e_m^{(j}(n-1) + \sum_{k=1}^{K} h_{m+1}^{(j,k)}\epsilon_\mu^{(k)}(n)$)	$JK(M-1)$
end	
for $i = 1$ to I do	
for $l = 0$ to $L-1$ do	
$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l)$	IJL
end	
end	
end	
Total: $2IJL + JKM + (2I+J)(M-1) + K$	

To compute the controller outputs, the multichannel equivalent of (16) is

$$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n) \tag{30}$$

where $r_m(n)$ in this case is defined as

$$r_m(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} x^{(i)}(n-l)x^{(i)}(n-l-m). \tag{31}$$

In analogy with (17), $r_m(n)$ can be recursively computed as

$$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}. \tag{32}$$

Similarly $e_m^{(j)}(n)$ has an update similar to that in (18), as given by

$$e_m^{(j)}(n) = \begin{cases} \sum_{k=1}^{K} h_1^{(j,k)}\epsilon_\mu^{(k)}(n), & \text{if } m = 1 \\ e_{m-1}^{(j)}(n-1) + \sum_{k=1}^{K} h_m^{(j,k)}\epsilon_\mu^{(k)}(n), & \text{if } 2 \le m \le M. \end{cases} \tag{33}$$

Collecting (24), (29), (30), (32), and (33), we obtain an alternative, equivalent implementation of the multichannel filtered–X LMS algorithm. Table I lists the operations of this implementation, along with the number of MAC's required to implement each operation. The algorithm employs $2IJL + JKM + (2I+J)(M-1) + K$ MAC's per iteration, and it requires $IJL + JKM + IL + (I+J+1)M + K - 1$ memory locations to implement.

Remark: This implementation of the multichannel filtered–X LMS adaptive controller modifies the adjoint LMS/CPFE adaptive controller by including the second summation on the RHS of (30) and the supporting updates for $e_m^{(j)}(n)$ and $r_m(n)$, respectively. Since $e_m^{(j)}(n)$ is of $O(\mu(n))$, the performance difference between the multichannel filtered–X and adjoint LMS/CPFE algorithms can only be expected to be significant for large stepsizes, a fact that has been pointed out in [17], [18]. Because the adjoint LMS/CPFE algorithm is a filtered-error technique with an approximate group delay of M samples in the update rule, however, its performance is often worse than that of the filtered–X LMS algorithm. Moreover, the complexity difference between the two algorithms is relatively insignificant for systems with a large number of channels, as will now be shown.

C. Complexity Comparisons

We now compare the computational and memory requirements of the original and fast implementations of the multichannel filtered–X LMS algorithm. In this comparison, we consider three different problem scenarios. Each scenario is defined by specific choices of the controller filter length L and plant model filter length M that might be appropriate for a particular type of noise or vibration control task. In each case, we present the quantities $R_F^{(C)}$ and $R_F^{(M)}$ that denote the ratios of the numbers of MAC's and memory locations,

TABLE II
COMPLEXITY COMPARISON, STANDARD, AND FAST MULTICHANNEL FILTERED–X LMS ALGORITHMS, $L = 50$, $M = 25$

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
1	2	2	0.7512	0.9900	0.7235	0.7765	4	6	4	0.3574	0.3974	0.3313	0.3348
1	3	2	0.7508	0.9502	0.6933	0.7301	2	6	6	0.3506	0.3906	0.3469	0.3515
1	4	2	0.7506	0.9302	0.6772	0.7054	2	8	6	0.3505	0.3865	0.3412	0.3446
1	6	2	0.7504	0.9101	0.6605	0.6797	2	8	8	0.3082	0.3359	0.3096	0.3123
2	2	2	0.6259	0.8055	0.6259	0.6559	4	8	8	0.2311	0.2495	0.2288	0.2303
2	3	2	0.6256	0.7654	0.5877	0.6085	8	8	8	0.1925	0.2063	0.1820	0.1828
2	4	2	0.6255	0.7453	0.5672	0.5832	4	8	12	0.1845	0.1972	0.1927	0.1937
2	6	2	0.6253	0.7252	0.5459	0.5568	4	12	12	0.1844	0.1949	0.1888	0.1895
1	4	4	0.5726	0.6752	0.5358	0.5523	4	16	12	0.1844	0.1938	0.1869	0.1874
1	6	4	0.5722	0.6635	0.5241	0.5353	8	8	12	0.1449	0.1544	0.1444	0.1449
2	3	3	0.5009	0.6025	0.4928	0.5085	8	8	16	0.1202	0.1274	0.1244	0.1248
1	4	6	0.5015	0.5733	0.4771	0.4888	8	12	16	0.1201	0.1257	0.1211	0.1214
3	3	3	0.4552	0.5424	0.4490	0.4601	8	16	16	0.1201	0.1249	0.1195	0.1197
3	6	3	0.4549	0.5130	0.4111	0.4168	8	32	16	0.1200	0.1236	0.1170	0.1171
2	4	4	0.4294	0.4979	0.4209	0.4305	16	16	16	0.1000	0.1036	0.0926	0.0927
2	6	4	0.4291	0.4862	0.4060	0.4125	16	32	16	0.1000	0.1024	0.0901	0.0901
2	8	4	0.4290	0.4804	0.3985	0.4033	8	16	32	0.0817	0.0842	0.0901	0.0903
4	4	4	0.3576	0.4090	0.3484	0.3536	16	16	32	0.0613	0.0631	0.0625	0.0625
2	4	6	0.3510	0.3989	0.3582	0.3651	32	32	32	0.0510	0.0520	0.0466	0.0466

TABLE III
COMPLEXITY COMPARISON, STANDARD, AND FAST MULTICHANNEL FILTERED–X LMS ALGORITHMS, $L = 2$, $M = 10$

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5472	1.0566	1.0682	1.1705	6	12	6	0.1901	0.2306	0.0.5900	0.5950
2	3	2	0.5443	0.9430	1.0667	1.1417	6	6	12	0.1663	0.1970	0.5398	0.5450
2	4	2	0.5429	0.8857	1.0658	1.1250	6	12	12	0.1653	0.1859	0.5274	0.5301
2	2	4	0.4902	0.7549	0.9315	0.9932	8	16	12	0.1307	0.1461	0.4570	0.4587
2	4	4	0.4851	0.6634	0.9173	0.9511	8	12	16	0.1245	0.1380	0.4433	0.4450
2	6	4	0.4834	0.6325	0.9119	0.9352	8	16	16	0.1243	0.1359	0.4401	0.4414
2	6	6	0.4631	0.5638	0.8525	0.8687	8	16	24	0.1178	0.1256	0.4227	0.4236
2	6	8	0.4527	0.5287	0.8209	0.8333	8	16	32	0.1146	0.1204	0.4138	0.4145
4	6	4	0.2824	0.3870	0.7576	0.7746	16	32	16	0.0723	0.0781	0.2937	0.2941
4	8	6	0.2586	0.3193	0.6933	0.7025	16	16	32	0.0625	0.0669	0.2707	0.2711
4	8	8	0.2468	0.2926	0.6625	0.6696	16	32	32	0.0623	0.0652	0.2669	0.2671

respectively, required by the fast implementation with respect to the numbers of MAC's and memory locations needed for the original implementation. For comparison, we also provide the corresponding ratios $R_A^{(C)}$ and $R_A^{(M)}$ for the adjoint LMS/CPFE algorithm [17], [18] with respect to the original filtered–X LMS algorithm. Since the adjoint LMS/CPFE algorithm equations are used within the fast implementation, we have that $R_A^{(C)} < R_F^{(C)}$ and $R_A^{(M)} < R_F^{(M)}$, although the two algorithms' requirements are similar for systems with a large number of channels.

The first situation considered is a broadband noise control task in which the controller and plant model filter lengths are $L = 50$ and $M = 25$, respectively. The ratio $L/M = 2$ offers a reasonable balance between the performance and the robustness of the controller for fixed hardware resources in many applications. Table II shows the complexity and memory ratios for the different cases considered. As can be seen for all of the cases considered, the number of multiplies required for the new implementation of the multichannel filtered–X LMS algorithm is less than that of the original algorithm, and this difference is significant for systems with a large number of channels. In fact, for an N-input, N-output, N-error system, the complexity of the new implementation is approximately 80% of the original implementation when $N = 2$, 40% of the original when $N = 4$, 20% of the original when $N = 8$, and 10% of the original when $N = 16$. In addition, the number of memory locations required by the new implementation is also reduced and is less than 10% of the original algorithm's memory requirements for $I = J = K = N = 16$. These savings are significant, as they allow a multichannel control system to be implemented on a much simpler hardware platform.

We now consider tasks in which $L = 2$ and $M = 10$. Such a situation is typical of narrowband noise control problems in which each input signal is a single sinusoid of a different frequency; thus, each channel of the controller is dedicated to one tonal component of the unwanted acoustic field. Table III lists the ratio of MAC's and memory locations for the two algorithms with respect to the original filtered–X LMS algorithm in this situation. As can be seen, except for systems with a small number of channels, the new implementation requires

TABLE IV
COMPLEXITY COMPARISON, STANDARD, AND FAST MULTICHANNEL FILTERED–X LMS ALGORITHMS, $L = 10$, $M = 20$

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5745	0.9787	0.9098	0.9877	6	12	6	0.2109	0.2442	0.3911	0.3940
2	3	2	0.5735	0.8886	0.8779	0.9331	6	6	12	0.1629	0.1886	0.3375	0.3405
2	4	2	0.5730	0.8434	0.8604	0.9032	6	12	12	0.1625	0.1796	0.3257	0.3273
2	2	4	0.4656	0.6832	0.7488	0.7956	8	16	12	0.1354	0.1482	0.2755	0.2764
2	4	4	0.4636	0.6092	0.7102	0.7350	8	12	16	0.1227	0.1341	0.2603	0.2613
2	6	4	0.4629	0.5844	0.6963	0.7131	8	16	16	0.1227	0.1324	0.2578	0.2585
2	6	6	0.4226	0.5057	0.6381	0.6499	8	16	24	0.1098	0.1163	0.2394	0.2398
2	6	8	0.4016	0.4648	0.6067	0.6158	8	16	32	0.1033	0.1082	0.2299	0.2303
4	6	4	0.3086	0.3937	0.5452	0.5560	16	32	16	0.0817	0.0865	0.1690	0.1692
4	8	6	0.2639	0.3138	0.4760	0.4818	16	16	32	0.0620	0.0656	0.1434	0.1436
4	8	8	0.2408	0.2787	0.4440	0.4485	16	32	32	0.0619	0.0644	0.1409	0.1410

only a fraction of the MAC's and memory locations used by the original implementation. Thus, the new implementation reduces the controller's hardware complexity in narrowband control situations as well.

The third problem scenario considered is a task in which $L = 10$ and $M = 20$. These choices are typical for noise and vibration control tasks in which the input signals are measured by physical sensors, but the primary goal of the controller is to attenuate a relatively few number of tonal components. Table IV lists the respective complexity and memory usage ratios for different cases. As in the previous cases, we find that the new implementation of the filtered–X LMS algorithm save computations and memory locations for systems with a large number of channels.

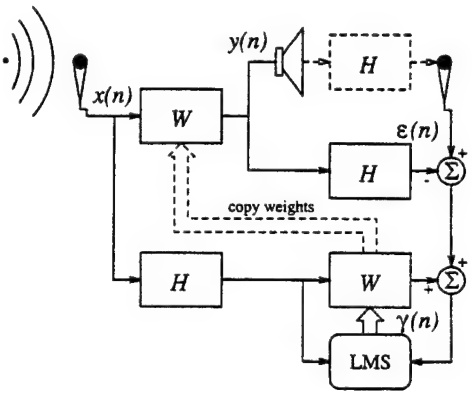

Fig. 2. Single-channel LMS adaptive controller employing delay compensation.

IV. LMS ALGORITHMS FOR ACTIVE NOISE CONTROL

A. Standard Implementation

In this section, we review the standard method for reducing the effects of the plant delay on the filtered–X LMS algorithm's operation and the resulting LMS algorithm for active noise control [20], [21]. Considering the single-channel filtered–X LMS adaptive controller, it is seen from (2) that the error signal $\epsilon(n)$ depends on the outputs $y(n-m)$ of the controller at different time instants, which in turn depend on the controller coefficients $w_l(n-m)$ at different time instants. Because the plant is typically causal, past coefficients are employed within the gradient-based updates, causing a decrease in the performance of the system not unlike that observed for the delayed LMS algorithm [31]. It is possible to largely mitigate the effects of this delay by computing a delay-compensated error signal that depends on the most-recent coefficients $w_l(n)$. Fig. 2 shows the block diagram of this system, in which $\gamma(k)$ is the delay-compensated error signal given by

$$\gamma(n) = \left\{ \epsilon(n) - \sum_{m=1}^{M} h_m y(n-m) \right\} + \sum_{l=0}^{L-1} w_l(n) f(n-l) \tag{34}$$

where the term within brackets on the RHS of (34) is nearly the same as the unattenuated noise signal $d(n)$ if the estimated impulse response h_m accurately models the unknown plant's impulse response. The LMS algorithm for active noise control uses $\gamma(k)$ to update the coefficients [20], [21] as

$$w_l(n+1) = w_l(n) - \mu(k)\gamma(k)f(n-l). \tag{35}$$

This algorithm requires a total of $3L + 2M + 1$ MAC's per iteration to implement, and it uses $2L+M+\max\{L, M+1\}+1$ memory locations. Note that this algorithm's performance depends on how well the estimated plant impulse response models the physical response of the plant. As our focus is on implementation and not performance issues, a performance analysis of the multichannel LMS algorithm for active noise control is beyond the scope of this paper.

We can easily extend the above algorithm to the multichannel case. In this situation, we compute the K delay-compensated error signals

$$\gamma^{(k)}(n) = \epsilon^{(k)}(n) - \sum_{j=1}^{J} \sum_{m=1}^{M} h_m^{(j,k)} y^{(j)}(n-m) + \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{l=0}^{L-1} w_l^{(i,j)}(n) f^{(i,j,k)}(n-l) \tag{36}$$

at which point $\mu(n)\gamma^{(k)}(n)$ is used in place of $\epsilon_\mu^{(k)}(n)$ in (23). Unfortunately, this modification adds $IJKL + JKM$ MAC's per iteration to the overall requirements of the adaptive system if the necessary $f^{(i,j,k)}(n)$ values are available, and it adds $IJKL + (2I+1)JKM$ MAC's if $f^{(i,j,k)}(n)$ must be

computed. In addition, the storage requirements for the overall system are significantly increased if the modification is applied to the fast multichannel filtered–X LMS algorithm in Table I.

B. New Implementations

1) A Multichannel Extension of an Existing Algorithm: In [23], a method is presented for reducing the complexity of the single-channel LMS algorithm for active noise control when the secondary path length M is less than a third of the controller filter length L. We now extend this algorithm to the multichannel case. Define

$$\begin{aligned}\gamma_{\mu}^{(n)}(n) &= \mu(n)\gamma^{(k)}(n) \\ &= \mu(n)\left\{\epsilon^{(k)}(n) + \sum_{j=1}^{J}\sum_{m=1}^{M} h_m^{(j,k)} u_m^{(j)}(n-1)\right\}\end{aligned} \tag{37}$$

where $u_m^{(j)}(n)$ is defined as

$$\begin{aligned}u_m^{(j)}(n) &= y^{(j)}(n-m) \\ &\quad - \sum_{i=1}^{I}\sum_{l=0}^{L-1} w_l^{(i,j)}(n+1)x^{(i)}(n-l-m).\end{aligned} \tag{38}$$

Using algebraic manipulations similar to those in [23], an update for $u_m^{(j)}(n)$ is found to be

$$u_m^{(j)}(n) = \begin{cases} -\sum_{k=1}^{K} \rho_0^{(j,k)}(n)\gamma_{\mu}^{(k)}(n), & \text{if } m = 1 \\ u_{m-1}^{(j)}(n-1) \\ \quad -\sum_{k=1}^{K} \rho_{m-1}^{(j,k)}(n)\gamma_{\mu}^{(k)}(n), & \text{if } 2 \le m \le M \end{cases} \tag{39}$$

where $\rho_m^{(j,k)}(n)$ is defined as

$$\rho_m^{(j,k)}(n) = \sum_{i=1}^{I}\sum_{l=0}^{L-1} x^{(i)}(n-l-m)f^{(i,j,k)}(n-l). \tag{40}$$

Note that $\rho_m^{(j,k)}(n)$ can be updated as

$$\begin{aligned}\rho_m^{(j,k)}(n) &= \rho_m^{(j,k)}(n-1) + \sum_{i=1}^{I}\Big\{x^{(i)}(n-m)f^{(i,j,k)}(n) \\ &\quad - x^{(i)}(n-m-L)f^{(i,j,k)}(n-L)\Big\}\end{aligned} \tag{41}$$

which greatly reduces the number of operations needed for the algorithm when L is large. This update also reduces the amount of memory required for the algorithm, as $f^{(i,j,k)}(n)$ and $f^{(i,j,k)}(n-L)$ can be computed at each iteration to avoid storing $f^{(i,j,k)}(n-l)$ for $1 \le l \le L-1$.

Collecting (37), (39), and (41), we obtain a multichannel delay compensation technique that requires $(4I+2)JKM$ MAC's per iteration and $(I+J+JK)M$ memory locations to implement when $L > M$, assuming that $f^{(i,j,k)}(n)$ and $f^{(i,j,k)}(n-L)$ are computed at each iteration. Comparing these complexity requirements with those of the original delay compensation technique, if

$$L > \left(2 + \frac{1}{I}\right)M \tag{42}$$

then this new technique is more computationally efficient. The new technique also has low memory requirements and thus is an ideal match to the fast algorithm in Table I.

2) An Alternate Implementation: Although useful, the delay-compensation method in (37), (39), and (41) can be prohibitive to implement when the number of channels is large, as its complexity grows as $O(IJKM)$. We now consider an alternate implementation that uses many of the existing quantities within the efficient multichannel filtered–X LMS algorithm in Table I while avoiding the formation of the filtered input signal values. For this derivation, consider the definition of $\rho_m^{(j,k)}(n)$ in (40). Substituting the expression for $f^{(i,j,k)}(n)$ in (22) into the RHS of (40) and rearranging terms, we obtain

$$\begin{aligned}&\rho_m^{(j,k)}(n) \\ &\quad = \sum_{q=1}^{M} h_q^{(j,k)}\left\{\sum_{l=0}^{L-1}\sum_{i=1}^{I} x^{(i)}(n-l-m)x^{(i)}(n-l-q)\right\} \\ &\quad = \sum_{q=1}^{M} h_q^{(j,k)} r_{m-q}(n-q)\end{aligned} \tag{43}$$

where $r_m(n)$ is as defined in (32). From the definition of $r_m(n)$, it is straightforward to show that

$$r_{m-q}(n-q) = r_{q-m}(n-m) \tag{44}$$

and thus the necessary values of $r_{m-q}(n-q)$ to represent $\rho_m^{(j,k)}(n)$ can be obtained from $r_m(n)$, $0 \le m \le M$ by storing delayed values of these quantities. Define

$$c_m^{(j)}(n) = \sum_{k=1}^{K} h_m^{(j,k)}(n)\gamma_{\mu}^{(k)}(n). \tag{45}$$

Note that $c_m^{(j)}(n)$ appears in the update for $e_m^{(j)}(n)$ in (33) when the delay compensation technique is combined with the fast filtered–X LMS algorithm; thus, it is already available. Then,

$$\sum_{k=1}^{K} \rho_{m-1}^{(j,k)}(n)\gamma_{\mu}^{(k)}(n) = \sum_{q=1}^{M} c_q^{(j)}(n) r_{m-1-q}(n-q) \tag{46}$$

and the RHS of (46) can replace the summations on the RHS of the updates for $u_m^{(j)}(n)$ in (39).

Table V lists the operations for this alternative form of the LMS algorithm for multichannel active noise control. This algorithm requires $JKM + J(M+1)(M-1)$ more MAC's per iteration than does the filtered–X LMS algorithm in Table I. If

$$M - \frac{1}{M} < (4I+1)K \tag{47}$$

then this implementation is more computationally efficient than that in (37), (39), and (41). If

$$M < IK + \sqrt{IKL + I^2K^2 + 1} \tag{48}$$

TABLE V
A MULTICHANNEL LMS ALGORITHM WITH REDUCED COMPLEXITY FOR ACTIVE NOISE CONTROL

Equation	*MACs*
for $m = 0$ to M do	
$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}$	$2I(M+1)$
end	
for $k = 1$ to K do	
$\gamma_\mu^{(k)}(n) = \mu(n) \left\{ e^{(k)}(n) + \sum_{j=1}^{J} \sum_{m=1}^{M} h_m^{(j,k)} u_m^{(j)}(n-1) \right\}$	$JKM + K$
end	
for $j = 1$ to J do	
$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n)$	$IJL + J(M-1)$
$c_1^{(j)}(n) = \sum_{k=1}^{K} h_1^{(j,k)} \gamma_\mu^{(k)}(n)$	JK
$e_1^{(j)}(n) = c_1^{(j)}(n)$	
for $m = 1$ to $M-1$ do	
$c_{m+1}^{(j)}(n) = \sum_{k=1}^{K} h_{m+1}^{(j,k)} \gamma_\mu^{(k)}(n)$	$JK(M-1)$
$e_{m+1}^{(j)}(n) = e_m^{(j)}(n-1) + c_{m+1}^{(j)}(n)$	JM
end	
$u_1^{(j)}(n) = -\sum_{q=1}^{M} c_q^{(j)}(n)r_q(n)$	JM
for $m = 1$ to $M-1$ do	
$u_{m+1}^{(j)}(n) = u_m^{(j)}(n-1) - \sum_{q=1}^{M} c_q^{(j)}(n)r_{m-q}(n-q)$	$JM(M-1)$
end	
for $i = 1$ to I do	
for $l = 0$ to $L-1$ do	
$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l)$	IJL
end	
end	
end	
Total: $2IJL + 2JKM + (2I + JM)(M+1) + K$	

then this implementation is more computationally efficient than the standard implementation in (36). Considering the system configurations listed in Tables II–IV, we find that the algorithm in Table V is the most computationally efficient method out of the three delay-compensation techniques considered when $IK \geq 7$, $IK \geq 5$, and $IK \geq 8$, respectively. For the remaining configurations, the standard delay-compensation implementation combined with the new filtered–X LMS update method in Table II is the most efficient, although the method in (37), (39), and (41) is the most efficient for the configurations in Table II if the controller filter length is increased to $L = 75$.

Remark: These implementations of the multichannel LMS adaptive controller modify the filtered–X LMS adaptive controller by including the summation within brackets on the RHS of (37) and the supporting updates for $u_m^{(j)}(n)$. Since $u_m^{(j)}(n)$ is of $O(\mu(n))$, the performance difference between the two multichannel LMS algorithms and the filtered–X LMS algorithm can only be expected to be significant for large stepsizes. Note that the filtered–X LMS algorithm is typically derived assuming "slow adaptation," so that the derivatives of the error signals with respect to the filter coefficients can be easily calculated [11]. Our multichannel LMS algorithms quantitatively define the difference between the filtered–X LMS and LMS coefficient updates and provide an alternative justification for the former algorithm for situations in which the stepsize is small-valued.

V. SIMULATIONS AND NUMERICAL ISSUES

In this section, we consider the effects that numerical errors due to finite precision calculations have on the performances of the new implementations of the filtered–X LMS and LMS algorithms for active noise control. One important feature of the LMS algorithm in adaptive filtering is its robust behavior in the presence of various approximations and errors that are often introduced in a real-world implementation. Since the original implementation of the filtered–X LMS algorithm and the adjoint LMS/CPFE algorithm are variants of stochastic gradient methods [16], they share many of the robust convergence properties of the LMS algorithm. The new implementations of the filtered–X LMS and LMS algorithms apply one or more forms of delay compensation to the adjoint LMS/CPFE algorithm. As such, the numerical properties of the delay compensation techniques are of immediate interest, particularly as they affect the long-term performances of the systems.

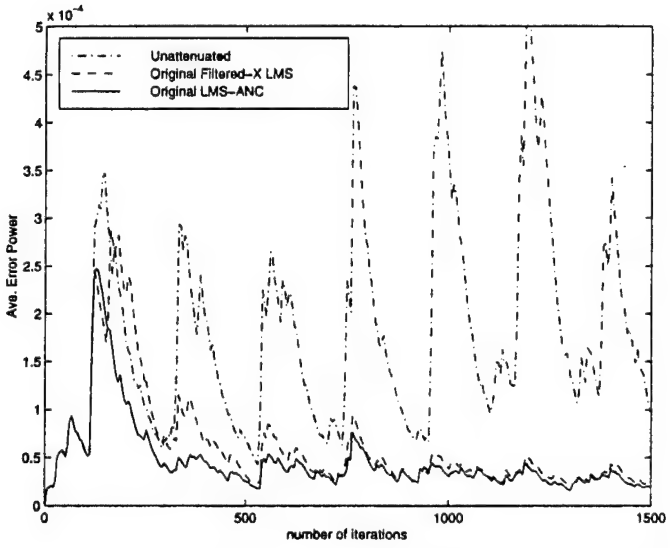

Fig. 3. Simulated performance on air compressor noise for the original filtered–X LMS and original LMS-ANC algorithms.

While formal analyses of the numerical properties of the delay compensation techniques used in our implementations are beyond the scope of this paper, extensive simulations of the implementations have indicated that the robust numerical properties of the underlying stochastic gradient algorithms are not fundamentally altered in our new implementations. These behaviors are quite unlike those of fast RLS/Kalman techniques that exhibit an exponential instability unless careful measures are taken [32], [33]. The only possible source of numerical difficulty is the method for calculating $r_m(n)$ in (32), as this update is marginally stable. Thus, numerical errors in $r_m(n)$ can grow linearly over time in a finite-precision environment, particularly in floating-point realizations in which relatively-few bits are allocated for the mantissas of the terms used to update each $r_m(n)$. Fortunately, the growth in these errors can be easily prevented using several well-known procedures. Perhaps the simplest procedure is to periodically recalculate $r_m(n)$ using its definition in (31), a procedure that requires extra additions and memory locations. Moreover, because each $r_m(n)$ has a finite memory by definition, accumulating and copying its value to the appropriate memory location within the controller causes no performance penalty, unlike periodic restart methods in exponentially windowed fast RLS/Kalman filters [32]. Another solution is to introduce a leakage factor into the calculation of $r_m(n)$. One particularly useful method, described in more detail in [34], is in (49), shown at the bottom of the page, where λ is slightly less than one. This method alters the value of $r_m(n)$ slightly, but for values of λ close to one, the errors introduced into the calculations for $y(n)$ do not significantly affect the overall behaviors of the respective systems. Moreover, the update in (49) adds only M MAC's and a single comparison to each systems' overall complexity.

Figs. 3–5 plot the envelope of the sum-of-squared errors $\sum_{k=1}^{K}\left\{\epsilon^{(k)}(n)\right\}^2$ for a total of seven different four-input, three-output, four-error active noise control algorithms with $L = M = 50$ as applied to air compressor data measured in an anechoic environment [35]. In this case, all calculations were performed in the MATLAB floating-point environment, and the approximate sampling rate of the data was 4 kHz. Stepsizes for each algorithm were chosen to provide the fastest convergence on this data while yielding approximately the same steady-state error power due to limits in noise modeling error. Fig. 3 shows the unattenuated air compressor noise signal, in which the bursty nature of the compressor noise is clearly evident, along with the average error power envelopes of the original filtered–X LMS and LMS algorithms applied to this data, in which the stepsizes for each algorithm were chosen as $\mu = 0.1$ and $\mu = 0.2$, respectively. Shown for comparison in Fig. 4 are the average error power envelopes of the adjoint LMS/CPFE algorithm, the fast filtered–X LMS algorithm in Table I, and the new multichannel LMS algorithm in Table V, in which the stepsizes for each algorithm were chosen as $\mu = 0.05$, $\mu = 0.1$, and $\mu = 0.2$, respectively. As can be seen, the fast multichannel filtered–X LMS algorithm outperforms the adjoint LMS/CPFE algorithm in its convergence rate, and the multichannel LMS algorithm performs the best of the three due to the lack of coefficient delay within the parameter updates. In addition, the differences in the error signals between the original and fast algorithms in Figs. 3 and 4 were found to be about ten times the order of the machine precision used in the simulation ($\approx 10^{-16}$) after 60 000 iterations. A linear growth of the numerical errors was apparent, however.

Shown in Fig. 5 are the behaviors of the fast multichannel filtered–X LMS and fast multichannel LMS algorithms in which the leakage-based update for $r_m(n)$ in (49) is employed, where $\lambda = 0.999$. Comparing the average error powers with those of the corresponding algorithms in Fig. 4, no discernible differences in performance can be seen. In fact, the actual differences between the errors of the corresponding systems were less than 2×10^{-10} in magnitude in this example—a negligible difference—and no growth in the numerical errors of $r_m(n)$ was observed. Thus, the method in (49) can be used to stabilize the marginal instability of the sliding-window $r_m(n)$ updates without altering the observed performances of the proposed systems.

VI. Conclusions

We have described new implementations of the multichannel filtered–X LMS and LMS algorithms for feedforward

$$r_m(n) = \begin{cases} \lambda\left\{r_m(n-1) - \sum_{i=1}^{I} x^{(i)}(n-L)x^{(i)}(n-L-m)\right\} + \sum_{i=1}^{I} x^{(i)}(n)x^{(i)}(n-m), & \text{if } n \bmod L = 0 \\ r_m(n-1) + \sum_{i=1}^{I} x^{(i)}(n)x^{(i)}(n-m) - \lambda\left\{\sum_{i=1}^{I} x^{(i)}(n-L)x^{(i)}(n-L-m)\right\}, & \text{otherwise} \end{cases} \tag{49}$$

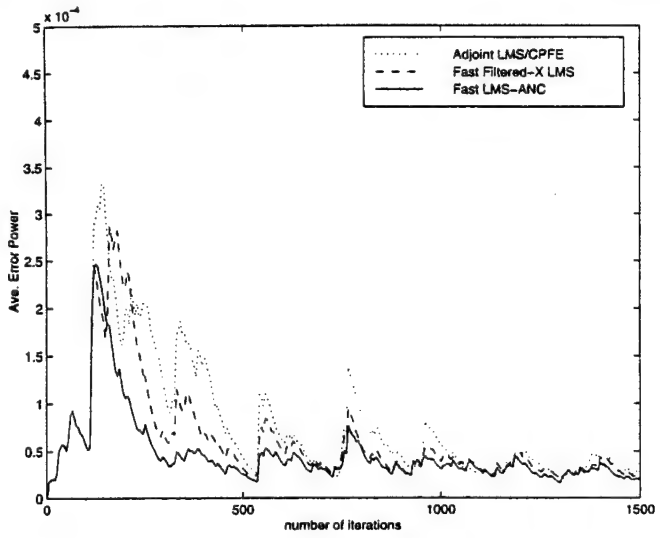

Fig. 4. Simulated performance on air compressor noise for the adjoint LMS/CPFE, fast filtered–X LMS, and fast LMS-ANC algorithms.

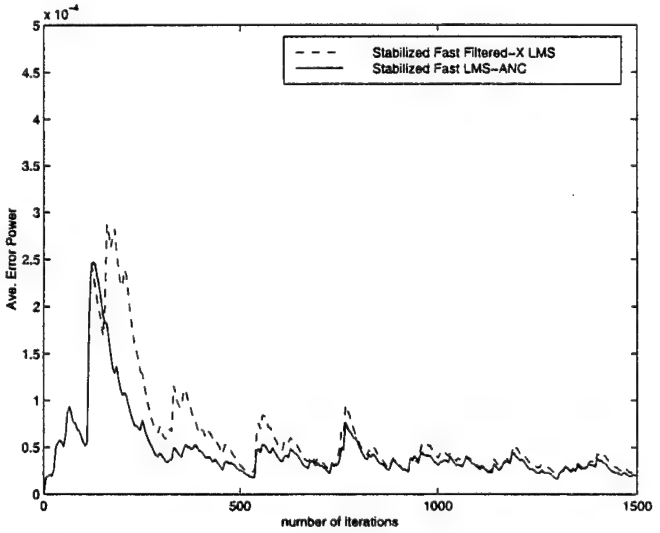

Fig. 5. Simulated performance on air compressor noise for the stabilized fast filtered–X LMS and stabilized fast LMS-ANC algorithms.

active noise and vibration control tasks. These implementations provide the same input–output behaviors of the original implementations while requiring only a fraction of the computational effort and memory of the original implementations. Because of the pervasiveness of stochastic-gradient-based algorithms for active noise and vibration control systems, the new implementations are expected to have a significant impact on the practicality and cost of these schemes in real-world applications.

ACKNOWLEDGMENT

The author would like to thank J. K. Soh and the numerous anonymous reviewers for helpful comments on this material. Data was provided by SRI International, Menlo Park, CA.

REFERENCES

[1] G. E. Warnaka, "Active attenuation of noise—The state of the art," *Noise Contr. Eng.*, vol. 18, pp. 100–110, May/June 1982.

[2] J. C. Stevens and K. K. Ahuja, "Recent advances in active noise control," *AIAA J.*, vol. 29, pp. 1058–1067, July 1991.

[3] S. J. Elliott and P. A. Nelson, "Active noise control," *IEEE Signal Processing Mag.*, vol. 10, pp. 12–35, Oct. 1993.

[4] C. R. Fuller and A. H. von Flotow, "Active control of sound and vibration," *IEEE Control Syst. Mag.*, vol. 15, pp. 9–19, Dec. 1995.

[5] E. F. Berkman and E. K. Bender, "Perspectives on active noise and vibration control," *Sound Vibrat.*, vol. 31, pp. 80–94, Jan. 1997.

[6] M. O. Tokhi and R. R. Leitch, *Active Noise Control*. Oxford, U.K.: Clarendon, 1992.

[7] P. A. Nelson and S. J. Elliott, *Active Control of Sound*. New York: Academic, 1992.

[8] S. M. Kuo and D. R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*. New York: Wiley, 1996.

[9] C. R. Fuller, S. J. Elliott, and P. A. Nelson, *Active Control of Vibration*. New York: Academic, 1996.

[10] P. Leug, "Process of silencing sound oscillations," U.S. Patent 2 0434 16, 1936.

[11] B. Widrow, D. Shur, and S. Shaffer, "On adaptive inverse control," in *Proc. 15th Asilomar Conf. Circuits, Systems, and Computers*, Pacific Grove, CA, Nov. 1981, pp. 185–195.

[12] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[13] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[14] Q. Shen and A. S. Spanias, "Time- and frequency-domain X–block least-mean-square algorithms for active noise control," *Noise Contr. Eng. J.*, vol. 44, pp. 281–293, Nov./Dec. 1996.

[15] S. C. Douglas, "Adaptive filters employing partial updates," *IEEE Trans. Circuits Syst.—II: Analog Digital Signal Processing*, vol. 44, pp. 209–216, Mar. 1997.

[16] B. Widrow and E. Walach, *Adaptive Inverse Control*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[17] S. R. Popovich, "Simplified parameter update for identification of multiple input, multiple output systems," in *Proc. Int. Congr. Noise Eng.*, Yokohama, Japan, Aug. 1994, vol. 2, pp. 1229–1232.

[18] E. A. Wan, "Adjoint LMS: An efficient alternative to the filtered–X LMS and multiple error LMS algorithms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, May 1996, vol. 3, pp. 1842–1845.

[19] E. Bjarnason, "Analysis of the filtered–X LMS algorithm," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 504–514, Nov. 1995.

[20] ——, "Active noise cancellation using a modified form of the filtered–X LMS algorithm," in *Proc. EUSIPCO'92, Signal Processing VI*, Brussels, Belgium, vol. 2, pp. 1053–1056.

[21] I. Kim, H. Na, K. Kim, and Y. Park, "Constraint filtered–X and filtered-U algorithms for the active control of noise in a duct," *J. Acoust. Soc. Amer.*, vol. 95, pp. 3397–3389, June 1994.

[22] M. Rupp, "Saving complexity of modified filtered–X LMS and delayed update LMS algorithms," *IEEE Trans. Circuits Syst.—II: Analog Dig. Signal Process.*, vol. 44, pp. 45–48, Jan. 1997.

[23] S. C. Douglas, "Efficient implementation of the modified filtered–X LMS algorithm," *IEEE Signal Processing Lett.*, vol. 4, pp. 286–288, Oct. 1997.

[24] S. C. Douglas and J. K. Soh, "Delay compensation methods for stochastic gradient adaptive filters," in *Proc. 8th IEEE DSP Workshop*, Bryce Canyon, UT, Aug. 1998, paper 108.

[25] S. D. Snyder and C. H. Hansen, "The effect of transfer function estimation errors on the filtered–X LMS algorithm," *IEEE Trans. Signal Processing*, vol. 42, pp. 950–953, Apr. 1994.

[26] S. C. Douglas, "Fast exact filtered–X LMS and LMS algorithms for multichannel active noise control," in *Proc. IEEE Int. Conf. Acoust.ics, Speech, Signal Processing*, Munich, Germany, Apr. 1997, vol. 1, pp. 399–402.

[27] ——, "Reducing the computational and memory requirements of the multichannel filtered–X LMS adaptive controller," in *Proc. Nat. Conf. Noise Control Engineering*, Philadelphia, PA, June 1997, vol. 2, pp. 209–220.

[28] ——, "Method and apparatus for multichannel active noise and vibration control," patent pending.

[29] S. Gay and S. Tavathia, "The fast affine projection algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, May 1995, vol. 5, pp. 3023–3026.

[30] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Detroit, MI, May 1995, vol. 2, pp. 945–948.

[31] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1397–1405, Sept. 1989.

[32] J. M. Cioffi and T. Kailath, "Fast recursive least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 304–337, Apr. 1984.

[33] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 92–114, Jan. 1991.

[34] S. C. Douglas and J. K. Soh, "A numerically-stable sliding-window estimator and its application to adaptive filters," in *Proc. 31st Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 1997, vol. 1, pp. 111–115.

[35] D. K. Peterson, W. A. Weeks, and W. C. Nowlin, "Active control of complex noise problems using a broadband multichannel controller," in *Proc. Nat. Conf. Noise Control Engineering*, Ft. Lauderdale, FL, May 1994, pp. 315–320.

Scott C. Douglas (S'88–M'92–SM'98) received the B.S. (with distinction), M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1988, 1989, and 1992, respectively.

From 1992 to 1998, he was an Assistant Professor in the Department of Electrical Engineering, University of Utah, Salt Lake City. Since August 1998, he has been with the Department of Electrical Engineering, Southern Methodist University, Dallas, TX, as an Associate Professor. He is a frequent consultant to industry in the areas of signal processing and adaptive filtering. His research activities include adaptive filtering, active noise control, blind deconvolution and source separation, and VLSI/hardware implementations of digital signal processing systems. He is the author or co-author of four book chapters and more than 80 articles in journals and conference proceedings; he also served as a Section Editor for *The Digital Signal Processing Handbook* (Boca Raton, FL: CRC, 1998). He has one patent pending.

Dr. Douglas received the Hughes Masters Fellowship Award in 1988 and the NSF Graduate Fellowship Award in 1989. He was a recipient of the NSF CAREER Award in 1995. He is currently an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE SIGNAL PROCESSING LETTERS, and the *Journal of VLSI Signal Processing Systems*. He is a member of both the Neural Networks for Signal Processing Technical Committee and the Education Technical Committee of the IEEE Signal Processing Society. He served on the Technical Program Committees of the 1995 IEEE Symposium on Circuits and Systems, Seattle, WA, and the 1998 IEEE Digital Signal Processing Workshop, Bryce Canyon, UT. He is the Proceedings Co-chair of the 1998 Workshop on Neural Networks for Signal Processing, Madison, WI, and is the Proceedings Editor of the 1999 International Symposium on Active Control of Sound and Vibration, Ft. Lauderdale, FL. He is the exhibits co-chair of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City. He is a member of Phi Beta Kappa and Tau Beta Pi.

EFFICIENT IMPLEMENTATIONS OF THE NLMS ALGORITHM WITH DECORRELATION FILTERS FOR ACOUSTIC ECHO CANCELLATION

J.K. Soh[1] *and S.C. Douglas*[2]

[1]Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA

[2]Department of Electrical Engineering, Southern Methodist University, Dallas, TX 75275 USA

ABSTRACT

The performance of the normalized least-mean-square (NLMS) adaptive filter in acoustic echo cancellation can be improved using decorrelation filters in the adaptation paths. This modification typically increases the complexity of the overall system by about 50%, however, and the system's memory requirements are also increased. In this paper, we provide four novel efficient implementations of the NLMS acoustic echo canceller with decorrelation filters whose complexities are nearly the same as that of the standard NLMS adaptive filter for short decorrelation filter lengths. Two of these implementations also save on memory use over the standard implementation. The algorithms are direct extensions of delay-compensation methods for feedforward active noise control systems.

1. INTRODUCTION

In acoustic echo cancellation for hands-free telephony, the impulse response of a loudspeaker-enclosure-microphone (LEM) system is estimated and used to electronically-remove the far-end speech component emitted by the loudspeaker from the near-end speech signal collected at the microphone. We can express the enhanced speech signal as

$$e(k) = y(k) - \widehat{y}(k), \tag{1}$$

where $y(k)$ is the microphone signal and $\widehat{y}(k)$ is an estimate of the far-end speech component within this signal. Typically, an adaptive finite-impulse-response (FIR) adaptive filter is used to calculate $\widehat{y}(k)$ as

$$\widehat{y}(k) = \mathbf{c}^T(k)\mathbf{x}(k), \tag{2}$$

where $\mathbf{c}(k) = [c_0(k) \ \cdots \ c_{N-1}(k)]^T$ is the estimated LEM impulse response at time k and $\mathbf{x}(k) = [x(k) \ \cdots \ x(k-N+1)]^T$ are the most-recent N samples of the far-end speech signal at the loudspeaker's output. The goal is to calculate $\mathbf{c}(k)$ such that the magnitude of $e(k)$ is minimized over time.

The normalized least-mean-square (NLMS) adaptive filter is perhaps the most-popular system for acoustic echo cancellation due to its simplicity and robust behavior. This adaptive filter is slow to converge in this application, however, due to the correlated nature of the far-end speech signal $x(k)$. One method to improve the behavior of NLMS in this application is shown in Fig. 1 [1]–[4], in which two decorrelation filters with impulse response $\underline{\mathbf{f}} = [f_0 \ \cdots \ f_{M-1}]^T$ are used to prefilter $x(k)$ and $y(k)$ as

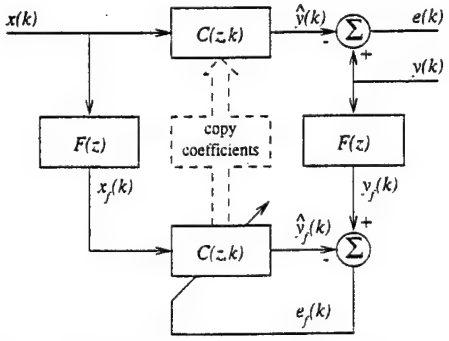

Fig 1: NLMS acoustic echo canceller employing decorrelation filters.

$$x_f(k) = \underline{\mathbf{f}}^T\underline{\mathbf{x}}(k) \quad \text{and} \quad y_f(k) = \underline{\mathbf{f}}^T\underline{\mathbf{y}}(k), \tag{3}$$

where $\underline{\mathbf{x}}(k) = [x(k) \ \cdots \ x(k-M+1)]^T$ and $\underline{\mathbf{y}}(k) = [y(k) \ \cdots \ y(k-M+1)]^T$, respectively. The values in $\underline{\mathbf{f}}$ are chosen such that $x_f(k)$ is approximately decorrelated; that is, $E\{x_f(k)x_f(k-l)\} \approx \delta(l)$, where $E\{\cdot\}$ denotes statistical expectation. Then, $\mathbf{c}(k)$ is updated as

$$\mathbf{c}(k+1) = \mathbf{c}(k) + \mu(k)\frac{e_f(k)}{\mathbf{x}_f^T(k)\mathbf{x}_f(k)}\mathbf{x}_f(k) \tag{4}$$

$$e_f(k) = y_f(k) - \widehat{y}_f(k) \tag{5}$$

$$\widehat{y}_f(k) = \mathbf{c}^T(k)\mathbf{x}_f(k), \tag{6}$$

where $\mathbf{x}_f(k) = [x_f(k) \cdots x_f(k-N+1)]^T$ and $\mu(k)$ is a step size in the range $0 < \mu(k) < 2$.

For typical speech signals, the above algorithm outperforms the standard NLMS algorithm in acoustic echo cancellation, and the improvement is significant even for fixed decorrelation filters as short as $M = 2$ taps [4]. As M can be much less than the echo canceller length N, the calculation of $x_f(k)$ and $y_f(k)$ in (3) do not add significant complexity to the overall system. Because $\widehat{y}(k)$ in (2) needs to be computed for the echo canceller's output, however, this algorithm requires about $3N + 2M$ multiply/accumulates (MACs) per iteration, which is more than a 50% increase in complexity over the standard NLMS algorithm that uses only $2N$ MACs per iteration. Moreover, this algorithm also requires more memory than the standard NLMS algorithm, as both $\mathbf{x}(k)$ and $\mathbf{x}_f(k)$ must be stored.

[0]This material is based upon work supported by the U.S. Army Research Office under Grant # DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the U.S. government, and no official endorsement should be inferred.

In this paper, we present four efficient implementations of the NLMS algorithm with decorrelation filters that are mathematically-equivalent to (1)–(6). The algorithms are direct extensions of delay-compensation methods for feedforward active noise control systems [5]–[8] that use correction terms to determine $\widehat{y}_f(k)$ in (6) from the value of $\widehat{y}(k)$ in (2). Each of the implementations only require $2N + O(M)$ MACs per iteration; as such, they are simpler than the standard implementation for $M \ll N$. Moreover, two of the implementations do not require storing $\mathbf{x}_f(k)$, thus reducing the memory requirements of the implementation as well. Tables of each of the implementations are provided, and overall comparisons of the algorithms' computational and memory requirements are given.

2. THE METHOD

We first outline the methodology behind the new computationally-efficient implementations. More details about these methods in the context of feedforward active noise control can be found in [5]–[8]. Define $\widehat{y}'_f(k)$ as

$$\widehat{y}'_f(k) = \underline{\mathbf{f}}^T \underline{\widehat{\mathbf{y}}}(k), \tag{7}$$

where $\underline{\widehat{\mathbf{y}}}(k) = [\widehat{y}(k) \cdots \widehat{y}(k-M+1)]^T$. Then, $\widehat{y}_f(k)$ in (6) can be calculated from $\widehat{y}'_f(k)$ as

$$\widehat{y}_f(k) = \widehat{y}'_f(k) + t(k), \tag{8}$$

where the correction term $t(k)$ is defined as

$$t(k) = \mathbf{c}^T(k)\mathbf{x}_f(k) - \underline{\mathbf{f}}^T \underline{\widehat{\mathbf{y}}}(k) \tag{9}$$

$$= \sum_{n=0}^{N-1} c_n(k) x_f(k-n) - \sum_{m=0}^{M-1} f_m \widehat{y}(k-m). \tag{10}$$

We can expand $t(k)$ in (10) using (3) and (4) as

$$t(k) = \sum_{i=1}^{M-1} f_i \sum_{j=0}^{N-1} x(k-j-i) \sum_{m=1}^{i} e_\mu(k-m) \cdot \sum_{n=0}^{M-1} f_n x(k-j-m-n), \tag{11}$$

where $e_\mu(k) = \mu(k) e_f(k) / \mathbf{x}_f^T(k)\mathbf{x}_f(k)$. By alternating the order of summations on the RHS of (11), we can obtain many different ways of computing $t(k)$, and as shown in [6], several of these are of $O(M)$ complexity per iteration. Thus, $\widehat{y}_f(k)$ can be computed from $\widehat{y}(k)$ using only $O(M)$ MACs per iteration, and the overall complexity of the system is therefore $2N + O(M)$ MACs per iteration.

3. COMPUTATIONALLY-EFFICIENT IMPLEMENTATIONS

The new algorithm implementations can be divided into two classes: those that use $\mathbf{x}_f(k)$ directly within the coefficient updates, and those that do not. The former class of algorithms, denoted as Class "C", are the simplest computationally, whereas the latter class, called Class "M", use somewhat less memory. Within each class, there are two implementations that are about the same complexity per iteration. We now derive these algorithms.

Table 1: Algorithm C1

Equation	*MACs*
$y_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{y}}(k)$	M
$\widehat{y}(k) = \mathbf{c}^T(k)\mathbf{x}(k)$	N
$e(k) = y(k) - \widehat{y}(k)$	1
$\widehat{y}_f(k) = \underline{\mathbf{f}}^T \underline{\widehat{\mathbf{y}}}(k) + \widetilde{\mathbf{f}}^T \overline{\mathbf{u}}(k-1)$	$2M-1$
$x_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{x}}(k)$	M
$\rho(k) = \rho(k-1) + \lvert x_f(k)\rvert^2 - \lvert x_f(k-N)\rvert^2$	2
$\overline{\mathbf{r}}_{xf}(k) = \overline{\mathbf{r}}_{xf}(k-1) + \overline{\mathbf{x}}(k)x_f(k) - \overline{\mathbf{x}}(k-N)x_f(k-N)$	$2M-2$
$e_\mu(k) = \mu(k)\left[y_f(k) - \widehat{y}_f(k)\right] / \rho(k)$	$1(1\div)$
$\overline{\mathbf{u}}(k) = \begin{bmatrix} 0 \\ \widetilde{\mathbf{u}}(k-1) \end{bmatrix} + e_\mu(k)\overline{\mathbf{r}}_{xf}(k)$	$M-1$
$\mathbf{c}(k+1) = \mathbf{c}(k) + e_\mu(k)\mathbf{x}_f(k)$	N
Total: $2N + 7M(1\div)$	

Algorithm C1: Rewrite $t(k)$ in (11) as

$$t(k) = \sum_{i=1}^{M-1} f_i \sum_{m=1}^{i} e_\mu(k-m) \cdot \sum_{j=0}^{N-1} x(k-j-i) x_f(k-j-m) \tag{12}$$

$$= \sum_{i=1}^{M-1} f_i \sum_{m=1}^{i} e_\mu(k-m) r_{xf,i-m}(k-m), \tag{13}$$

where we have defined

$$r_{xf,i}(k) = \sum_{j=0}^{N-1} x(k-j-i) x_f(k-j). \tag{14}$$

By defining $\overline{\mathbf{r}}_{xf}(k) = [r_{xf,0}(k) \cdots r_{xf,M-2}(k)]^T$ and $\overline{\mathbf{u}}(k) = [u_1(k) \cdots u_{M-1}(k)]^T$ where

$$u_i(k) = \sum_{m=1}^{i} e_\mu(k-m+1) r_{xf,i-m}(k-m+1), \tag{15}$$

we can express (13) as

$$t(k) = \sum_{i=1}^{M-1} f_i u_i(k-1) = \widetilde{\mathbf{f}}^T \overline{\mathbf{u}}(k-1), \tag{16}$$

where $\widetilde{\mathbf{f}} = [f_1 \cdots f_{M-1}]^T$ contains the last $M-1$ elements of $\underline{\mathbf{f}}$. Furthermore, it can be easily shown that

$$\overline{\mathbf{u}}(k) = \begin{bmatrix} 0 \\ \widetilde{\mathbf{u}}(k-1) \end{bmatrix} + e_\mu(k)\overline{\mathbf{r}}_{xf}(k) \tag{17}$$

$$\overline{\mathbf{r}}_{xf}(k) = \overline{\mathbf{r}}_{xf}(k-1) + \overline{\mathbf{x}}(k)x_f(k) - \overline{\mathbf{x}}(k-N)x_f(k-N) \tag{18}$$

where $\widetilde{\mathbf{u}}(k)$ contains the first $M-2$ elements of $\overline{\mathbf{u}}(k)$ and $\overline{\mathbf{x}}(k) = [x(k) \cdots x(k-M+2)]^T$. From (16)–(18), we can compute $\widehat{y}_f(k)$ in (8) from $\widehat{y}(k)$ using only about $5M$ MACs at each time step. Table 1 lists the entire algorithm; its overall complexity is $2N + 7M$ MACs and one divide per iteration.

Table 2: Algorithm C2

Equation	*MACs*
$y_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{y}}(k)$	M
$\widehat{y}(k) = \mathbf{c}^T(k)\mathbf{x}(k)$	N
$e(k) = y(k) - \widehat{y}(k)$	1
$\widehat{y}_f(k) = \underline{\mathbf{f}}^T \widehat{\underline{\mathbf{y}}}(k) + \overline{\mathbf{p}}^T(k-1)\overline{\mathbf{e}}(k-1)$	$2M-1$
$x_f(k) = x^{(1)}(k-1) + f_0 x(k)$	1
$x_f^{(M)}(k) = x_f^{(M)}(k-N) = 0$	0
for $j = M-1$ to 1 step (-1) do	
$\quad x_f^{(j)}(k) = x_f^{(j+1)}(k) + f_j x(k-j+1)$	$M-1$
$\quad x_f^{(j)}(k-N) = x_f^{(j+1)}(k-N) + f_j x(k-j-N+1)$	$M-1$
$\quad p_j(k) = p_j(k-1) + x_f(k-j+1)x_f^{(j)}(k) - x_f(k-j-N+1)x_f^{(j)}(k-N)$	$2M-2$
end	
$\rho(k) = \rho(k-1) + \lvert x_f(k)\rvert^2 - \lvert x_f(k-N)\rvert^2$	2
$e_\mu(k) = \mu(k)\left[y_f(k) - \widehat{y}_f(k)\right]/\rho(k)$	$1(1\div)$
$\underline{\mathbf{e}}(k) = \begin{bmatrix} e_\mu(k) \\ \overline{\mathbf{e}}(k-1) \end{bmatrix}$	0
$\mathbf{c}(k+1) = \mathbf{c}(k) + e_\mu(k)\mathbf{x}_f(k)$	N
Total: $2N + 7M(1\div)$	

Algorithm C2: Define $\overline{\mathbf{p}}(k) = [p_1(k) \ \cdots \ p_{M-1}(k)]^T$ where

$$p_m(k) = \sum_{j=0}^{N-1} x_f(k-j-m+1) \sum_{i=m}^{M-1} f_i x(k-j-i+1). \quad (19)$$

Then, it can be shown that [6]

$$t(k) = \sum_{m=1}^{M-1} p_m(k-1)e_\mu(k-m) = \overline{\mathbf{p}}^T(k-1)\overline{\mathbf{e}}(k-1), \quad (20)$$

where $\overline{\mathbf{e}}(k) = [e_\mu(k) \ \cdots \ e_\mu(k-M+2)]^T$. Furthermore, by defining

$$x_f^{(m)}(k) \ = \ \sum_{i=m}^{M-1} f_i x(k-i+1), \quad (21)$$

for $0 \le m \le M-1$, $p_m(k)$ and $x_f^{(m)}(k)$ can be updated in an order-recursive fashion as

$$p_m(k) \ = \ p_m(k-1) + x_f(k-m+1)x_f^{(m)}(k) - x_f(k-m-N+1)x_f^{(m)}(k-N) \quad (22)$$

$$x_f^{(m)}(k) \ = \ \begin{cases} f_{M-1}x(k-M) & \text{if } m = M-1 \\ x_f^{(m+1)}(k) + f_m x(k-m+1) & \text{if } m < M-1. \end{cases} \quad (23)$$

From (20), (22), and (23), we can again compute $y_f(k)$ in (8) from $y(k)$ using about $5M$ MACs per iteration. Table 2 lists the operations of this implementation that, like Algorithm C1, requires only $2N+7M$ MACs and one divide per iteration to implement.

Algorithms M1 and M2: Both Algorithms C1 and C2 use the filtered input signal vector $\mathbf{x}_f(k)$ to update $\mathbf{c}(k)$ at each time instant. W now develop two alternative equivalent updating strategies that employ the input signal vector $\mathbf{x}(k)$ for coefficient updates. These methods combine the adjoint LMS/CPFE algorithm in [9, 10], a delay-compensation technique used in the fast affine projection (FAP) algorithm [11, 12], and the procedures outlined previously. For derivations of similar algorithms used in active noise control, see [7, 8].

Table 3: Algorithm M1

Equation	*MACs*
$y_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{y}}(k)$	M
$\widetilde{\mathbf{r}}_{xx}(k) = \widetilde{\mathbf{r}}_{xx}(k-1) + \overline{\mathbf{x}}(k-1)x(k) - \overline{\mathbf{x}}(k-N-1)x(k-N)$	$2M-2$
$\widehat{y}(k) = \widehat{\mathbf{c}}^T(k)\mathbf{x}(k) + \widetilde{\mathbf{r}}_{xx}^T(k)\overline{\mathcal{E}}(k-1)$	$N+M-1$
$e(k) = y(k) - \widehat{y}(k)$	1
$\widehat{y}_f(k) = \underline{\mathbf{f}}^T \widehat{\underline{\mathbf{y}}}(k) + \widetilde{\mathbf{f}}^T \overline{\mathbf{u}}(k-1)$	$2M-1$
$x_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{x}}(k)$	M
$x_f(k-N) = \underline{\mathbf{f}}^T \underline{\mathbf{x}}(k-N)$	M
$\rho(k) = \rho(k-1) + \lvert x_f(k)\rvert^2 - \lvert x_f(k-N)\rvert^2$	2
$\overline{\mathbf{r}}_{xf}(k) = \overline{\mathbf{r}}_{xf}(k-1) + \overline{\mathbf{x}}(k)x_f(k) - \overline{\mathbf{x}}(k-N)x_f(k-N)$	$2M-2$
$e_\mu(k) = \mu(k)\left[y_f(k) - \widehat{y}_f(k)\right]/\rho(k)$	$1(1\div)$
$\underline{\mathcal{E}}(k) = \begin{bmatrix} 0 \\ \overline{\mathcal{E}}(k-1) \end{bmatrix} + \underline{\mathbf{f}} e_\mu(k)$	M
$\overline{\mathbf{u}}(k) = \begin{bmatrix} 0 \\ \widetilde{\mathbf{u}}(k-1) \end{bmatrix} + e_\mu(k)\overline{\mathbf{r}}_{xf}(k)$	$M-1$
$\widehat{\mathbf{c}}(k+1) = \widehat{\mathbf{c}}(k) + \epsilon_{M-1}(k)\mathbf{x}(k-M+1)$	N
Total: $2N + 12M - 3(1\div)$	

Define an auxiliary coefficient vector $\widehat{\mathbf{c}}(k)$ as

$$\widehat{\mathbf{c}}(k) \ = \ \mathbf{c}(k) - \overline{\mathbf{X}}(k-1)\overline{\underline{\mathcal{E}}}(k-1), \quad (24)$$

where $\overline{\mathbf{X}}(k)$ contains the first $M-1$ columns of the $(N \times M)$ matrix $\mathbf{X}(k) = [\mathbf{x}(k) \ \cdots \mathbf{x}(k-M+1)]$ and $\overline{\mathcal{E}}(k)$ contains the first $M-1$ elements of $\underline{\mathcal{E}}(k) = [\epsilon_0(k) \ \cdots \ \epsilon_{M-1}(k)]^T$, defined as

$$\underline{\mathcal{E}}(k) \ = \ \begin{bmatrix} \varepsilon_0(k) \\ \varepsilon_0(k-1) + \varepsilon_1(k) \\ \vdots \\ \varepsilon_0(k-M+1) + \cdots + \varepsilon_{M-1}(k) \end{bmatrix} \quad (25)$$

$$\varepsilon_m(k) \ = \ f_m e_\mu(k). \quad (26)$$

It is easily shown that $\widehat{\mathbf{c}}(k)$ in (24) has the update

$$\widehat{\mathbf{c}}(k+1) \ = \ \widehat{\mathbf{c}}(k) + \epsilon_{M-1}(k)\mathbf{x}(k-M+1). \quad (27)$$

In addition, $\underline{\mathcal{E}}(k)$ can be updated using

$$\underline{\mathcal{E}}(k) \ = \ \begin{bmatrix} 0 \\ \overline{\mathcal{E}}(k-1) \end{bmatrix} + \underline{\mathbf{f}} e_\mu(k). \quad (28)$$

The output $\widehat{y}(k)$ can be calculated from the auxiliary coefficient vector and a properly-defined correction term. Premultiplying both sides of (24) by $\mathbf{x}^T(k)$ gives the relation

$$\widehat{y}(k) \ = \ \widehat{\mathbf{c}}^T(k)\mathbf{x}(k) + \mathbf{x}^T(k)\overline{\mathbf{X}}(k-1)\overline{\underline{\mathcal{E}}}(k-1). \quad (29)$$

Define $\widetilde{\mathbf{r}}_{xx}(k) = [r_1(k) \ \cdots \ r_{M-1}(k)]^T$, where $r_m(k)$ is

$$r_m(k) \ = \ \sum_{n=0}^{N-1} x(k-n-m)x(k-n). \quad (30)$$

Then, we can rewrite (29) as

$$\widehat{y}(k) \ = \ \widehat{\mathbf{c}}^T(k)\mathbf{x}(k) + \widetilde{\mathbf{r}}_{xx}^T(k)\overline{\mathcal{E}}(k-1), \quad (31)$$

Table 4: Algorithm M2

Equation	*MACs*
$y_f(k) = \underline{\mathbf{f}}^T \underline{\mathbf{y}}(k)$	M
$\widetilde{\mathbf{r}}_{xx}(k) = \widetilde{\mathbf{r}}_{xx}(k-1) + \overline{\mathbf{x}}(k-1)x(k) - \overline{\mathbf{x}}(k-N-1)x(k-N)$	$2M-2$
$\widehat{y}(k) = \widehat{\mathbf{c}}^T(k)\mathbf{x}(k) + \widetilde{\mathbf{r}}_{xx}^T(k)\overline{\mathcal{E}}(k-1)$	$N+M-1$
$e(k) = y(k) - \widehat{y}(k)$	1
$\widehat{y}_f(k) = \underline{\mathbf{f}}^T \underline{\widehat{\mathbf{y}}}(k) + \overline{\mathbf{p}}^T(k-1)\overline{\mathbf{e}}(k-1)$	$2M-1$
$x_f(k) = x^{(1)}(k-1) + f_0 x(k)$	1
$x_f(k-N) = x^{(1)}(k-N-1) + f_0 x(k-N)$	1
$x_f^{(M)}(k) = x_f^{(M)}(k-N) = 0$	0
for $j = M-1$ to 1 step (-1) do	
$\quad x_f^{(j)}(k) = x_f^{(j+1)}(k) + f_j x(k-j+1)$	$M-1$
$\quad x_f^{(j)}(k-N) = x_f^{(j+1)}(k-N) + f_j x(k-j-N+1)$	$M-1$
$\quad p_j(k) = p_j(k-1) + x_f(k-j+1)x_f^{(j)}(k) - x_f(k-j-N+1)x_f^{(j)}(k-N)$	$2M-2$
end	
$\rho(k) = \rho(k-1) + \lvert x_f(k)\rvert^2 - \lvert x_f(k-N)\rvert^2$	2
$e_\mu(k) = \mu(k)\left[y_f(k) - \widehat{y}_f(k)\right]/\rho(k)$	$1(1\div)$
$\underline{\mathcal{E}}(k) = \begin{bmatrix} 0 \\ \overline{\mathcal{E}}(k-1) \end{bmatrix} + \underline{\mathbf{f}} e_\mu(k)$	M
$\underline{\mathbf{e}}(k) = \begin{bmatrix} e_\mu(k) \\ \overline{\mathbf{e}}(k-1) \end{bmatrix}$	0
$\widehat{\mathbf{c}}(k+1) = \widehat{\mathbf{c}}(k) + \epsilon_{M-1}(k)\mathbf{x}(k-M+1)$	N
Total: $2N + 11M - 2(1\div)$	

and $\widetilde{\mathbf{r}}_{xx}(k)$ can be computed recursively as

$$\widetilde{\mathbf{r}}_{xx}(k) = \widetilde{\mathbf{r}}_{xx}(k-1) + \overline{\mathbf{x}}(k-1)x(k) - \overline{\mathbf{x}}(k-N+1)x(k-N). \quad (32)$$

Combining (27), (28), (31), and (32), we can compute $\widehat{y}(k)$ and update $\widehat{\mathbf{c}}(k)$ without forming the vector $\mathbf{x}_f(k)$. Embedding this technique within Algorithm C1 and Algorithm C2, one obtains Algorithm M1 and M2 listed in Tables 3 and 4, respectively. These algorithms require $2N + 12M - 3$ and $2N + 11M - 2$ MACs per iteration to implement, respectively.

4. SUMMARY AND CONCLUSIONS

In summary, we have presented four mathematically-equivalent implementations of the NLMS acoustic echo canceller with decorrelating filters in the adaptation loop. These algorithms are straightforward extensions of recently-developed delay-compensation methods for feedforward active noise control systems. To enable a quick comparison, Table 5 lists the number of MACs and memory locations that each algorithm requires, ignoring constant terms. From this table we see that

- if $N > 5M$, Algorithms C1 and C2 require fewer MACs as compared to the standard implementation at the expense of slight increases in memory, and
- for $N > 10M$, Algorithms M1 and M2 require fewer MACs and fewer memory locations as compared to the standard implementation.

In addition, we can make the following remarks:

Remark #1: Simulations indicate that the proposed algorithms enjoy the superior numerical properties shared by other stochastic gradient adaptive filters. The only source of numerical difficulties appear to be the marginally-stable sliding-window updates for $\rho(k)$, $\overline{\mathbf{r}}_{xf}(k)$, and $\widetilde{\mathbf{r}}_{xx}(k)$. If numerical error accumulation in these updates is a problem, they can be periodically-restarted with only a small increase in memory, or some form of leakage can be introduced [7].

Table 5: System Requirements

Algorithm	*Complexity (MACs)*	*Memory Requirements*
Standard	$3N + 2M$	$3N + 2M$
Algorithm C1	$2N + 7M$	$3N + 6M$
Algorithm C2	$2N + 7M$	$3N + 7M$
Algorithm M1	$2N + 12M$	$2N + 8M$
Algorithm M2	$2N + 11M$	$2N + 10M$

Remark #2: The above algorithms have been derived assuming that the decorrelation filter impulse response $\underline{\mathbf{f}}$ is time-invariant. If $\underline{\mathbf{f}}$ is time-varying, then these implementations are not guaranteed to produce a sequence $\widehat{y}(k)$ that is identical to that of the standard implementation. Note that it is possible to both compute $\widehat{y}(k)$ *and* implicitly update the coefficient vector as

$$\mathbf{c}(k+1) = \mathbf{c}(k) + e_{\mu,k}(k)\mathbf{x}_{f,k}(k) \quad (33)$$

$$e_{\mu,k}(k) = \frac{\mu(k)}{\mathbf{x}_{f,k}^T(k)\mathbf{x}_{f,k}(k)}\left[\underline{\mathbf{f}}^T(k)\underline{\mathbf{y}}(k) - \mathbf{c}^T(k)\mathbf{x}_{f,k}(k)\right] \quad (34)$$

$$\mathbf{x}_{f,k}(k) = \mathbf{X}(k)\underline{\mathbf{f}}(k) \quad (35)$$

by an algorithm that is of $2N + O(M^2)$ complexity. Partial details of such an algorithm are given in [6, Table 1]. Moreover, if $\underline{\mathbf{f}}(k)$ is updated by an algorithm with a shift-input regressor vector, this complexity can be further reduced. Additional details regarding the forms and performances of these algorithms are deferred to a future publication.

REFERENCES

[1] S. Yamamoto, S. Kitayama, J. Tamura, and H. Ishigami, "An adaptive echo canceller with linear predictor," *Trans. Inst. Electron. Commun. Engr. Japan*, vol. E62, no. 12, pp. 851-857, Dec. 1979.

[2] R. Frenzel and M.E. Hennecke, "Using prewhitening and step-size control to improve the performance of the LMS algorithm for acoustic echo compensation," *Proc. IEEE Int. Symp. Circuits Syst.*, San Diego, CA, vol. 4, pp. 1930-1932, 1992.

[3] T. Schertler and G. U. Schmidt, " Implementation of a low-cost acoustic echo canceller," *Proc. 5th Int. Workshop Acoustic Echo Noise Control*, London, UK, pp. 49-52, 1997.

[4] C. Breining, P. Dreiseitel, E. Hänsler, A. Mader, B. Nitsch, H. Puder, T. Schertler, G. Schmidt, and J. Telp, "Acoustic echo control," *IEEE Signal Processing Mag.*, vol. 16, no. 4, pp. 42-69, July 1999.

[5] S.C. Douglas, "Efficient implementation of the modified filtered-X LMS algorithm," *IEEE Signal Processing Lett.*, vol. 4, pp. 286-288, Oct. 1997.

[6] S.C. Douglas and J.K. Soh, "Delay compensation methods for stochastic gradient adaptive filters," *Proc. 9th IEEE Digital Signal Processing Workshop*, Bryce Canyon, UT, paper no. 108, Aug. 1998.

[7] S.C. Douglas, "Fast implementations of the filtered-X LMS and LMS algorithms for multichannel active noise control," *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 454-465, July 1999.

[8] S.C. Douglas, "Method and apparatus for multichannel active noise and vibration control," patent pending.

[9] S.R. Popovich, "Simplified parameter update for identification of multiple input, multiple output systems," *Proc. Int. Congr. Noise Eng.*, Yokohama, Japan, vol. 2, pp. 1229-1232, Aug. 1994.

[10] E.A. Wan, "Adjoint LMS: An efficient alternative to the filtered-X LMS and multiple error LMS algorithms," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, vol. 3, pp. 1842-1845, May 1996.

[11] S. Gay and S. Tavathia, "The fast affine projection algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 5, pp. 3023-3026, May 1995.

[12] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Detroit, MI, vol. 2, pp. 945-948, May 1995.

Rejection of disturbances with a large sinusoidal component of unknown frequency

Marc Bodson and Scott C. Douglas
Department of Electrical Engineering, University of Utah
Salt Lake City, UT 84112
(801) 581 8590 and *(801) 581 4445*
bodson@ee.utah.edu and *douglas@ee.utah.edu*

Abstract

An important application of smart materials and structures is the control of periodic disturbances or vibrations in environments such as aircrafts and helicopters. In these cases, the source of the noise is a rotating machine, so that a large component of the disturbance is periodic. While it is often possible to take measurements on the machine that is the source of the periodic disturbance, concerns of reliability and maintainability sometimes make such measurements undesirable, if not impossible. Then, the problem is to attenuate a periodic disturbance whose frequency is unknown. An adaptive algorithm is presented in this paper for periodic disturbance attenuation, using the concept of a phase-locked loop. For simplicity, the disturbance is assumed to be sinusoidal. An approximate analysis is performed and the results are found useful to select the design parameters. Simulations are presented that demonstrate the ability of the algorithm to reject sinusoidal disturbances with unknown frequency, and to follow signals with slowly varying magnitude and frequency. The effect of measurement noise and of additional disturbances is also analyzed. The results provide numerical measures of the parameter variations and of the loss of performance in the presence of noise.

Keywords: adaptive control, signal processing, periodic disturbances, phase-locked loops.

1. Introduction

The market for control systems that actively abate unwanted noise and vibration is rapidly expanding. Often, the source of the noise is a rotating machine such as a propeller engine in an aircraft [1, 2] or a fan engine in a ventilation system [3]. In some applications, active control is useful to cancel both vibration and noise, such as in helicopters [4] - [6]. With rotating machines, the disturbance that needs to be rejected contains a large sinusoidal component, and possibly a few harmonics. The period of the disturbance is usually not known exactly and may even vary during the system's operation. The disturbance rejection problem is more complicated when measurements cannot be taken on the source or close to it, because of practical considerations (such as in a headset), or because of reliability.

In the paper, we present an adaptive control algorithm for the rejection of sinusoidal disturbances with unknown frequency. An approximate, linearized analysis provides a simple representation of the closed-loop system that is useful for design, and simulations validate the theoretical results. The analysis is then extended to the non-ideal case where either measurement noise or additional disturbances are present. Again, the results prove useful for design and are verified in simulations. Work is in progress to extend the scheme to periodic, non-sinusoidal disturbances.

2. Rejection of periodic disturbances

2.1 Previous work

In previous work [7]-[10], an adaptive feedforward control (AFC) algorithm is studied for the purpose of cancelling disturbances with known frequency. The basic structure of the algorithm is shown in Fig. 1. The disturbance is assumed to be of the form

$$d(t) = \theta_1^* \cos(\alpha(t)) + \theta_2^* \sin(\alpha(t)). \tag{1}$$

Two adaptive parameters, θ_1 and θ_2, represent the estimates of the two components of the disturbance d, which is to be cancelled by the input u. $P(s)$ is the transfer function of the plant and $\alpha(t) = \int \omega(t)dt = \omega_1 t$, where ω_1 is the angular

0-8194-2090-5/96/$6.00

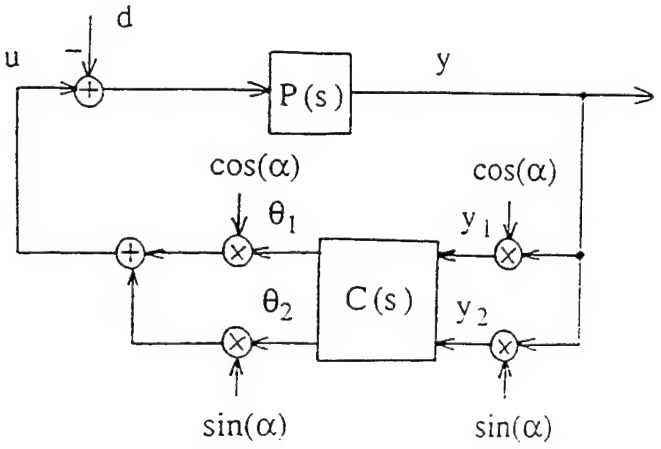

Figure 1: Adaptive feedforward cancellation scheme

frequency of the disturbance. For the design of the compensator $C(s)$, algorithms are available from standard adaptive control theory. A Lyapunov-type algorithm yields the following update laws for the adaptive parameters

$$\dot{\theta}_1 = -gy\cos(\alpha), \qquad \dot{\theta}_2 = -gy\sin(\alpha), \tag{2}$$

which is equivalent to letting the transfer function matrix of the compensator in Fig. 1 be $C(s) = -g$ diag $(1/s)$, where $g > 0$ is the adaptation gain. In [7], the equivalence between the adaptive feedforward control algorithm with update (2) and a linear time-invariant operator is established. The result can be used to explain a phenomenon observed in experiments where high-order harmonics are reduced, although the algorithm is not designed for this task. This analysis provides considerable insight on how to adjust the adaptive gains for stability and performance. The result has been used to design feedforward control algorithms exploiting the freedom in the design of the matrix $C(s)$. In [8], experiments applying the algorithm to control the position of a read/write head of a high-density 1 Gbyte disk drive are provided. The experimental results confirm the validity of the theoretical analysis and its usefulness for design. In fact, the maximum adaptation gain for stability can be predicted precisely, based on measurements of the open-loop frequency response. The closed-loop disturbance rejection properties of the algorithm have also been found experimentally to closely match those predicted by the theory. In [9] and [10], further refinements to the method are described so that loop dynamics can be shaped for optimal performance. Experimental results again support the conclusions of the analysis.

Averaging methods can be applied to approximate the responses of the adaptive systems. Defining the parameter errors $\varphi_1 = \theta_1 - \theta_1^*$ and $\varphi_2 = \theta_2 - \theta_2^*$, the averaged system is found to be

$$\frac{d}{dt}\begin{bmatrix} \varphi_{1,av} \\ \varphi_{2,av} \end{bmatrix} = -g\, G \begin{bmatrix} \varphi_{1,av} \\ \varphi_{2,av} \end{bmatrix}, \tag{3}$$

where

$$G = \frac{1}{2}\begin{bmatrix} P_R & P_I \\ -P_I & P_R \end{bmatrix}; \quad P_R = Re[P(j\omega_1)]; \quad P_I = Im[P(j\omega_1)]. \tag{4}$$

If the update laws (2) are replaced by

$$\frac{d}{dt}\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = -g\, G^{-1} \begin{bmatrix} y\cos(\alpha) \\ y\sin(\alpha) \end{bmatrix}, \tag{5}$$

the averaged system becomes

$$\frac{d}{dt}\begin{bmatrix} \varphi_{1,av} \\ \varphi_{2,av} \end{bmatrix} = -g \begin{bmatrix} \varphi_{1,av} \\ \varphi_{2,av} \end{bmatrix}. \tag{6}$$

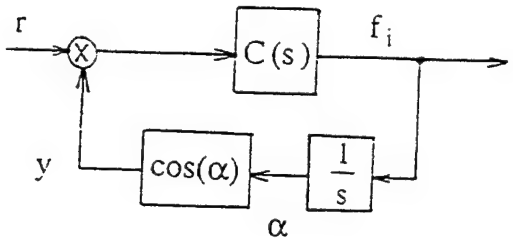

Figure 2: Phase-locked loop

This system is a set of simple first-order systems, whose dynamics are set by the adaptive gain $g > 0$. The scheme can be easily extended to the case where multiple sinusoids are present by adding terms to the control input and updating the coefficients in a similar manner as before.

2.2 Periodic disturbances with unknown frequency

In the application to disk drives described in [8], it was assumed that the period of the disturbance was known exactly, a reasonable assumption as the period of the disturbance is tied to the rotation of the disk, and synchronization pulses can be provided to the digital signal processing chip implementing the control computations. In other applications, it is often possible to place an encoder or a tachometer on the rotating machine that is at the origin of the disturbance. However, there are also many cases where it is impossible to place such sensors, or where the addition of a sensor implies an unacceptable cost in terms of reliability, as such sensors may be subjected to strong vibrations, high temperatures, or dirty environments.

The problem of cancelling periodic disturbances with unknown frequency has received less attention. When it is desired to filter a periodic noise component with unknown frequency from a signal, adaptive notch filters can perform the required operation (see, *e.g.*, [11], [12], [13]). However, notch filters are designed to solve filtering problems, not control problems. Some active noise and vibration control schemes use either a microphone placed ahead of the speakers that are used for noise attenuation or an accelerometer placed on the rotating machine that is the source of the vibration. While these are control problems, the disturbance is measured *before* it enters the plant, making the problem considerably simpler. We discuss here algorithms that are applicable in cases that do not satisfy such an assumption. Active noise control using a headset, for example, can be performed without external sound measurement.

In [14], we describe an adaptive algorithm for the attenuation of sinusoidal disturbances with unknown frequency. The approach extends the adaptive feedforward control (AFC) scheme of Fig. 1 by integrating a *phase-locked loop* within this scheme. In this paper, we first review the algorithm, then extend the analysis to the case where noise or additional disturbances are present.

3. Adaptive algorithm

The scheme under consideration uses a phase-locked loop structure commonly found in communication systems [15]. For reference, a basic phase-locked loop is shown on Fig. 2. A signal $r(t)$ with unknown frequency is applied to the input of the system. The signal $r(t)$ is multiplied by a signal $y(t)$ that is reconstructed by the phase-locked loop, and the result is filtered by the transfer function $C(s)$. This transfer function is typically the cascade of a low-pass filter and a proportional integral (PI) control law. The signal y is obtained by applying the output signal f_i to what is commonly called a *voltage-controlled oscillator*. In mathematical terms, this is the cascade of an integrator and a sinusoidal function. Ideally, the signals $r(t)$ and $y(t)$ are 90^0 out of phase; thus their average is zero, leading to a constant output signal. Since the output signal is the derivative of the phase of the input signal, it is also the instantaneous frequency of that signal.

At this point, it would be straightforward to apply the phase-locked loop structure to the task of estimating the instantaneous frequency of a sinusoidal signal. However, a more interesting application of the concept is in an integrated scheme for disturbance rejection. The challenges of such an integration are that, in a phase-locked loop, only the phase of the signals are matched and not their magnitude. Further, the input and output signals are multiplied in Fig. 2, and not subtracted as in Fig. 1.

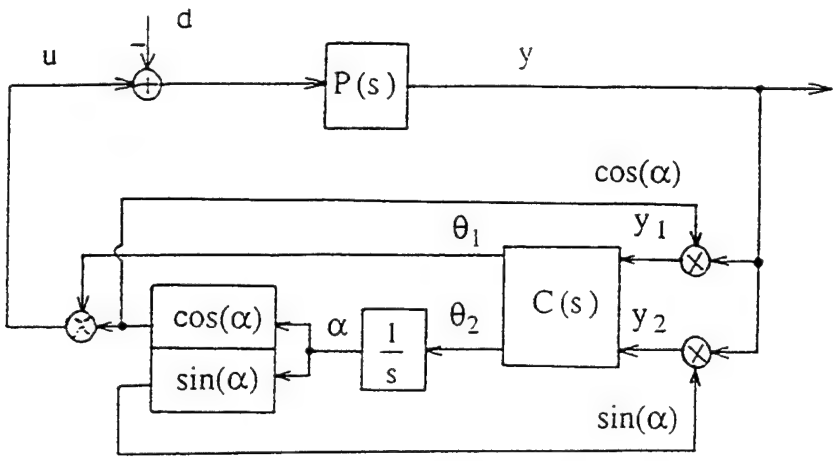

Figure 3: Adaptive algorithm for sinusoidal disturbance cancellation

Our scheme solves these problems and is shown in Fig. 3. In the figure, θ_1 is the estimate of the magnitude of the disturbance signal, and θ_2 is the estimate of its instantaneous frequency. Further, α is the estimate of the phase of the disturbance signal; it is the integral of the estimate of the frequency θ_2. The transfer function matrix $C(s)$ plays a similar role as in Fig. 1, but its design is different, and more complex. An approximate analysis of this system provide criteria for the design of $C(s)$.

4. Approximate analysis and compensator design

The analysis technique is similar to one used for analyzing phase-locked loops in FM communication systems [15]. We assume that the input and output of the plant are signals whose spectrum is concentrated around the frequency ω_1. In particular, this implies that the instantaneous frequency $\dot{\alpha} = \theta_2$ is close to ω_1. The analysis is based on the following assumptions:

- the disturbance has a single sinusoidal component $d(t) = d_1 \cos(\omega_1 t + \delta_1)$, where d_1, ω_1, and δ_1 are unknown;
- θ_1 and θ_2 vary sufficiently slowly that the response of the plant to the signal u can be considered to be the same as that of a sinusoidal input with frequency $\dot{\alpha} = \theta_2$;
- the instantaneous frequency $\dot{\alpha}$ is close to ω_1, so that $P(j\dot{\alpha})$ can be replaced by $P(j\omega_1)$;
- the phase error $(\alpha - \omega_1 t - \delta_1)$ is small.

With these assumptions, the output of the plant is approximately given by

$$y = -P_R d_1 \cos(\omega_1 t + \delta_1) + P_I d_1 \sin(\omega_1 t + \delta_1) + P_R \theta_1 \cos(\alpha) - P_I \theta_1 \sin(\alpha). \tag{7}$$

Two signals used by the algorithm are

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y\cos(\alpha) \\ -y\sin(\alpha) \end{bmatrix}. \tag{8}$$

Keeping only the low frequency part of these signals, the two components of y are approximately given by (using (7))

$$y_1 = -\frac{1}{2} P_R d_1 \cos(\omega_1 t + \delta_1 - \alpha) + \frac{1}{2} P_I d_1 \sin(\omega_1 t + \delta_1 - \alpha) + \frac{1}{2} P_R \theta_1,$$

$$y_2 = \frac{1}{2} P_R d_1 \sin(\omega_1 t + \delta_1 - \alpha) - \frac{1}{2} P_I d_1 \cos(\omega_1 t + \delta_1 - \alpha) + \frac{1}{2} P_I \theta_1. \tag{9}$$

Simulations in the next section show that the approximation is indeed excellent in the cases under consideration without introducing an additional lowpass filter.

With a small phase error $\alpha - \omega_1 t - \delta_1$, (9) gives the linearized relationship

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = G \begin{bmatrix} \theta_1 - d_1 \\ d_1(\alpha - \omega_1 t - \delta_1) \end{bmatrix}, \tag{10}$$

where G is the same matrix as defined in (4). Two variables x_1, x_2 are defined through

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = G^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}. \tag{11}$$

In terms of the parameter errors $\varphi_1 = \theta_1 - d_1$ and $\varphi_2 = \theta_2 - \omega_1$, they satisfy

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varphi_1 \\ d_1 \left(\int_0^t \varphi_2(\sigma) d\sigma - \delta_1 \right) \end{bmatrix}. \tag{12}$$

As a result, the dynamics of the system from the parameter errors φ_1 and φ_2 to the variables x_1 and x_2 are those of a unity gain and of an integrator with gain d_1, respectively. The compensator $C(s)$ can then be chosen to be the cascade of the transformation (11) and the standard adaptive laws of the form

$$\dot{\theta}_1 = \dot{\varphi}_1 = C_1(s)[x_1], \qquad \dot{\theta}_2 = \dot{\varphi}_2 = C_2(s)[x_2], \tag{13}$$

where $C_1(s)$ and $C_2(s)$ are selected to guarantee the closed-loop stability of the two systems. In the case of $C_1(s)$, the dynamics are those of a single integrator. In the case of $C_2(s)$, they are those of a double integrator. Therefore, one can choose

$$C_1(s) = -g_1, \qquad C_2(s) = -g_2 \frac{s+a}{s+b}, \tag{14}$$

where g_1, g_2, a, and b are parameters to be selected in order to yield a desirable closed-loop response. In summary, the adaptive algorithm is given by (8), (11), (13), (14), and

$$u = \theta_1 \cos(\alpha), \quad \dot{\alpha} = \theta_2. \tag{15}$$

Because the magnitude of the disturbance d_1 acts as a gain in the second transfer function, the parameters of the compensator $C_2(s)$ must be designed for a range of magnitudes of the disturbance d_1. Alternatively, simple techniques can be used to estimate the disturbance level before the algorithm is turned on.

The algorithm requires the knowledge of the matrix G whose elements depends on the unknown frequency ω_1. We can approximate the implementation of this system in two ways. In the first method, we design a compensator that works satisfactorily for a range of matrices G, corresponding to a range of frequencies. Fortunately, the uncertainty in both the frequency estimate θ_2 and the plant characteristics $P(j\omega_1)$ are often small enough such that a compensator designed for a frequency in the mid-range of the band of interest is satisfactory. Another option is to use the matrix corresponding to the estimated frequency $\dot{\alpha} = \theta_2$ for G^{-1}. The dynamics of the system are far more difficult to analyze in that case, but we have found this form of the algorithm to work well in simulations. Finally, note that whichever implementation method is chosen, some knowledge of the frequency response of the plant in the frequency range of interest is required.

Simulation results – Noiseless case

We let $P(s) = 100/(s+100)$, $\omega_1 = 100$, and $d_1 = 1$. The parameter g_1 is set to 10, leading to a closed-loop pole for the first system at -10 rad/s. The other parameters are set to $g_2 = 400$, $a = 5$, and $b = 30$, which lead to three closed-loop poles for the second system located at -10 rad/s and $-10 \pm j10$ rad/s.

Fig. 4 shows the output of the plant, in raw value on the left and in log scale on the right. The output is found to decrease to negligible values in less than a second. The adaptive parameters are shown on Fig. 5 and converge rapidly. On the left and on the right are the responses for θ_1 (magnitude) and for θ_2 (frequency) respectively. The smooth curves are the responses of the approximate system, while the oscillatory ones are for the original parameters. The match between the original and the approximate system responses is excellent for the frequency, and less so for the magnitude,

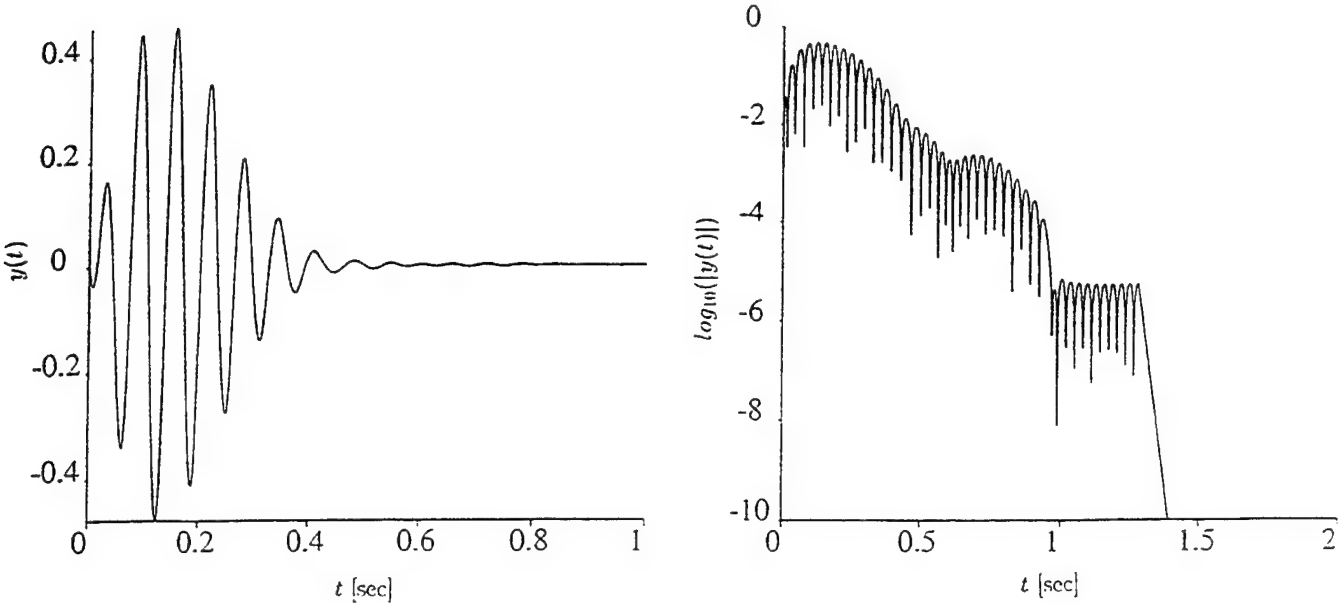

Figure 4: Plant output (left) and log of the plant output (right) – Noiseless case

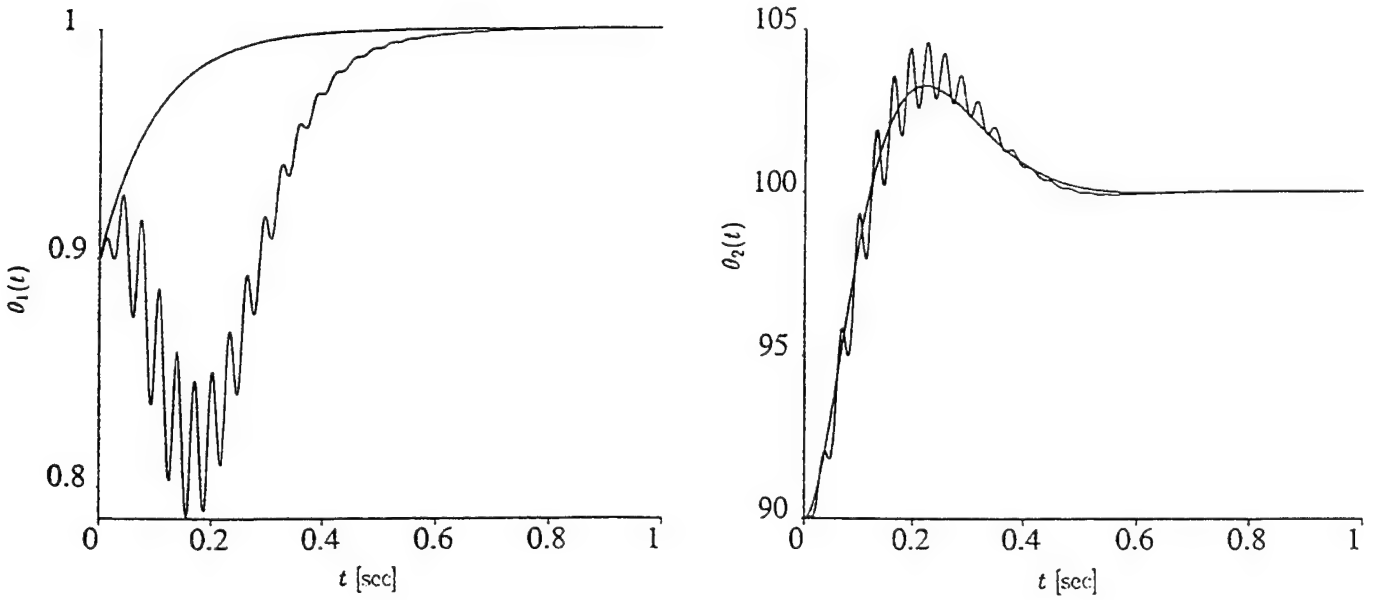

Figure 5: Adaptive parameters – Noiseless case

probably because of nonlinear effects that were neglected in the linearized analysis. Note, for example, that both $\theta_1 = 1$ and $\theta_1 = -1$ are potential equilibrium points of the algorithm.

In the simulation, the system was able to converge despite a 10% initial error in frequency. It was found that the scheme was able to acquire frequencies with errors up to 30%. It is well-known that an important characteristic of phase-locked loops is their *lock-in* (or *pull-in)* range [16]. Careful design of the loop transfer function can increase the lock-in range and improve performance. Additional supervisory logic can also be used for acquisition, and would be part of a practical design.

5. Noise analysis

The previous derivation considered the noiseless case, which resulted in nominal values for the parameters of $\theta_1^* = d_1$, $\theta_2^* = \omega_1$, and nominal functions $\alpha^*(t) = \omega_1 t$ and $y^*(t) = 0$. In this section, the perturbations due to a small noise δn added at the output is evaluated. We define $\theta_1 = \theta_1^* + \delta\theta_1$, $\theta_2 = \theta_2^* + \delta\theta_2$, $y = y^* + \delta y$, and $\bar{y} = y^* + \delta y + \delta n$, where y is the actual output of the plant and $\bar{y}$ is the measured output (the one used by the algorithm). Since $y^* = 0$, we see that $y = \delta y$ and $\bar{y} = \delta\bar{y}$, where $\delta\bar{y} = \delta y + \delta n$. We define the components of the measured output as

$$\begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \end{bmatrix} = \begin{bmatrix} \bar{y}\cos(\alpha) \\ -\bar{y}\sin(\alpha) \end{bmatrix}. \tag{16}$$

Again, these two variables are the ones that are used by the algorithm, instead of y_1 and y_2, which are defined for the purpose of analysis only. For the noise, we introduce the representation

$$\delta n(t) = \delta n_i(t)\cos(\alpha^*(t)) - \delta n_q(t)\sin(\alpha^*(t)), \tag{17}$$

where $\delta n_i(t)$ and $\delta n_q(t)$ are assumed to vary much slower than $\cos(\alpha^*(t))$. With the low-pass approximation, the noise components δn_1 and δn_2 are defined as

$$\begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix} = \begin{bmatrix} \delta n\cos(\alpha^*) \\ -\delta n\sin(\alpha^*) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\delta n_i \\ \frac{1}{2}\delta n_q \end{bmatrix}. \tag{18}$$

Note that there are two transfer functions of interest in the presence of noise: the transfer function from δn to y, and the transfer function from δn to $\bar{y}$. The first transfer function indicates how much the actual output is affected by the noise. One would want to make this transfer function small if δn represented measurement noise. However, the same formalism can also be used if δn represents the effect of additional disturbances that affect the output of the plant. In that case, one would want to make the second transfer function, from δn to $\bar{y}$, small, so that the measured output would be small. There may be cases where there are disturbances that one may not wish to cancel. One such case is in a noise control system in which the noise contains a large periodic component and a white noise component. The white noise may be considered acceptable physiologically, and would best be treated as measurement noise.

We now begin the analysis, noting that

$$\bar{y}_1 = \cos(\alpha^* + \delta\alpha)(y^* + \delta y + \delta n), \qquad \bar{y}_2 = -\sin(\alpha^* + \delta\alpha)(y^* + \delta y + \delta n), \tag{19}$$

so that, with $y^* = 0$ and neglecting second-order effects, it follows that

$$\delta\bar{y}_1 = \delta y\cos(\alpha^*) + \delta n_1, \qquad \delta\bar{y}_2 = -\delta y\sin(\alpha^*) + \delta n_2. \tag{20}$$

The next part of the analysis concerns the part of the algorithm that is linear, and therefore remains identical in terms of variations. Specifically, we have:

$$\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} = G^{-1}\begin{bmatrix} \delta\bar{y}_1 \\ \delta\bar{y}_2 \end{bmatrix}, \qquad \delta\theta_1 = \frac{C_1(s)}{s}\delta x_1, \qquad \delta\theta_2 = \frac{C_2(s)}{s}\delta x_2, \qquad \delta\alpha = \frac{1}{s}\delta\theta_2. \tag{21}$$

The control input and its variation are given by

$$u = (\theta_1^* + \delta\theta_1)\cos(\alpha^* + \delta\alpha), \qquad \delta u = \cos(\alpha^*)\delta\theta_1 - \theta_1^*\sin(\alpha^*)\delta\alpha. \tag{22}$$

Assuming slow variations of $\delta\theta_1$ and $\delta\alpha$, the output of the plant corresponding to this input is given by

$$\delta y = P_R\cos(\alpha^*)\delta\theta_1 - P_R\theta_1^*\sin(\alpha^*)\delta\alpha - P_I\sin(\alpha^*)\delta\theta_1 - P_I\theta_1^*\cos(\alpha^*)\delta\alpha. \tag{23}$$

Multiplication of this signal by $\cos(\alpha^*)$ and $\sin(\alpha^*)$ yield signals whose low-frequency components are given by (again neglecting second-order terms):

$$\delta\bar{y}_1 = \frac{1}{2}P_R\delta\theta_1 - \frac{1}{2}P_I\theta_1^*\delta\alpha + \delta n_1, \qquad \delta\bar{y}_2 = \frac{1}{2}P_R\theta_1^*\delta\alpha + \frac{1}{2}P_I\delta\theta_1 + \delta n_2. \tag{24}$$

As a result, we have that

$$\begin{bmatrix} \delta\bar{y}_1 \\ \delta\bar{y}_2 \end{bmatrix} = G \begin{bmatrix} \delta\theta_1 \\ \theta_1^*\delta\alpha \end{bmatrix} + \begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix}. \tag{25}$$

For the parameter variations, we have

$$\begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \end{bmatrix} = \begin{bmatrix} \frac{C_1(s)}{s} & 0 \\ 0 & \frac{C_2(s)}{s} \end{bmatrix} \left(\begin{bmatrix} \delta\theta_1 \\ \frac{\theta_1^*}{s}\delta\theta_2 \end{bmatrix} + G^{-1} \begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix} \right), \tag{26}$$

which leads to the following transfer function between the components of the noise and the parameter variations

$$\begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \end{bmatrix} = \begin{bmatrix} \frac{C_1(s)}{s - C_1(s)} & 0 \\ 0 & \frac{sC_2(s)}{s^2 - \theta_1^* C_2(s)} \end{bmatrix} G^{-1} \begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix}. \tag{27}$$

The relationship between the noise variations and the measured plant output variations is

$$\begin{bmatrix} \delta\bar{y}_1 \\ \delta\bar{y}_2 \end{bmatrix} = \left(G \begin{bmatrix} \frac{C_1(s)}{s - C_1(s)} & 0 \\ 0 & \frac{s\theta_1^* C_2(s)}{s^2 - \theta_1^* C_2(s)} \end{bmatrix} G^{-1} + I \right) \begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix}, \tag{28}$$

while the relationship to the actual plant output variations is

$$\begin{bmatrix} \delta y_1 \\ \delta y_2 \end{bmatrix} = G \begin{bmatrix} \frac{C_1(s)}{s - C_1(s)} & 0 \\ 0 & \frac{s\theta_1^* C_2(s)}{s^2 - \theta_1^* C_2(s)} \end{bmatrix} G^{-1} \begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix}. \tag{29}$$

The formulas can be useful in several ways. For example, if the noise δn has a flat power spectral density $S_n(f)$ over a bandwidth f_B centered around $f_1 = \omega_1/2\pi$, it is common to assume that $S_{n_i}(f)$ and $S_{n_q}(f)$ also have flat spectral density over a bandwidth f_B with the same power level but centered around $f = 0$ [15]. One can then derive the densities $S_{n_1}(f)$ and $S_{n_2}(f)$ as well as $S_{y_1}(f)$ and $S_{y_2}(f)$ using (29). The expected values of y_1^2 and y_2^2 can be calculated by integrating the power spectra. Finally, one concludes that the output y is a sinusoid with a randomly and slowly-varying peak, whose expected value is

$$E[y_i^2 + y_q^2] = 4\ E[y_1^2] + 4\ E[y_2^2]. \tag{30}$$

A similar calculation applies if the noise is deterministic, as will be shown in the next section.

Simulation results –Noisy case

The second set of simulations verifies the results of the analysis for the non-ideal case. Although the results can be used to assess the impact of measurement noise on the system, we elect to apply the results to study the effect of plant disturbances. We consider the case of deterministic disturbances, so that the results of the analysis can be precisely verified without extensive Monte-Carlo simulations. We also consider two special cases of additive output disturbances that correspond to an equivalent input disturbance that is a sinusoid with a varying magnitude, or with a varying instantaneous frequency. Interestingly, these two problems can be cast in the framework of additive output disturbances, and the results help to explain the tracking ability of the algorithm.

First, we assume that the disturbance $d(t)$ is replaced by $d(t) = (1 + 0.1\cos(t))\cos(100t)$, *i.e.*, a sinusoidal function that is amplitude modulated by a sinusoidal function of much lower frequency, with a modulation index of 10%. We take

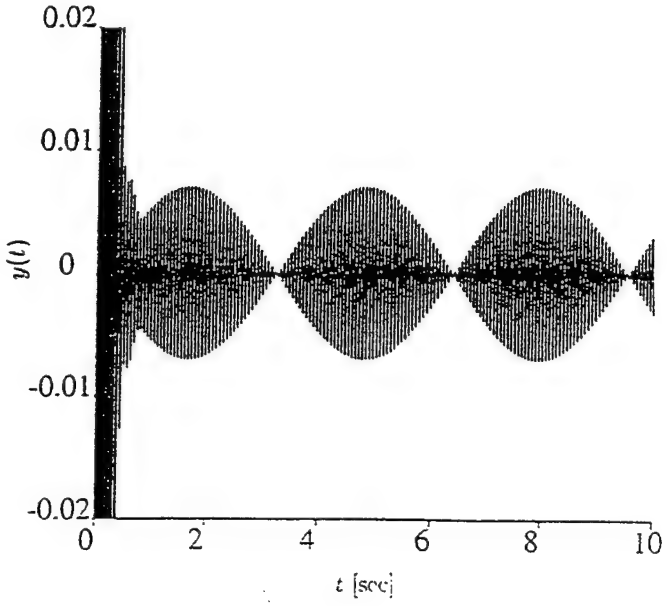

Figure 6: Plant output – Amplitude modulation of low frequency

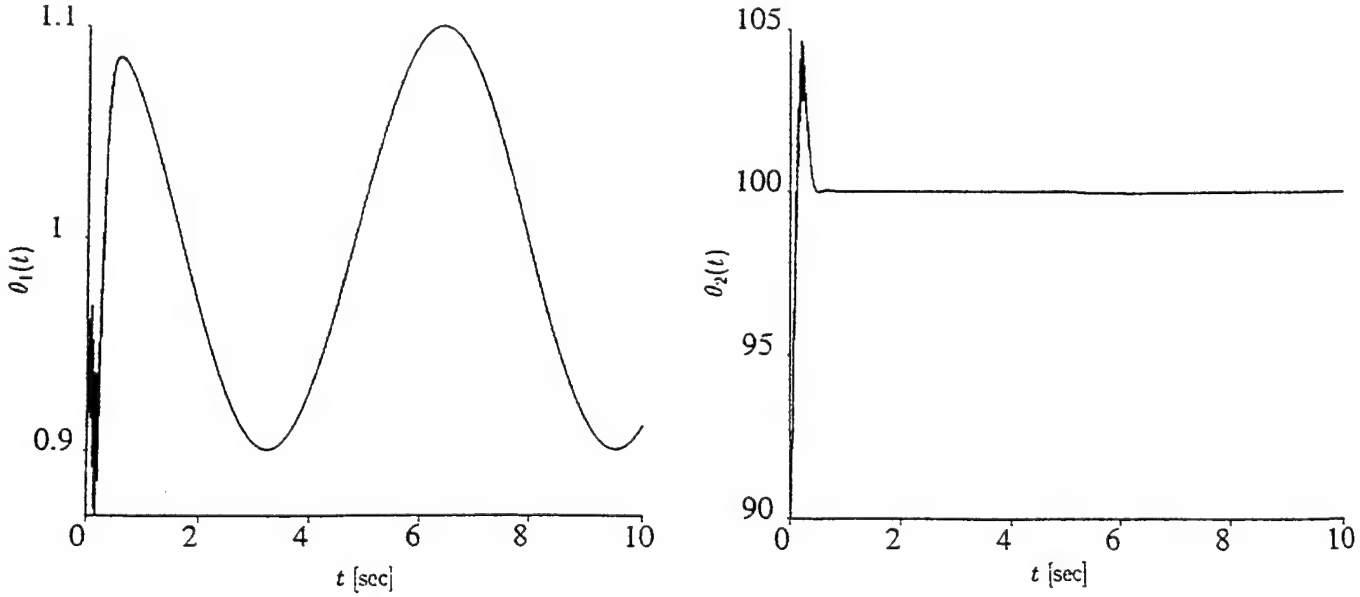

Figure 7: Adaptive parameters – Amplitude modulation of low frequency

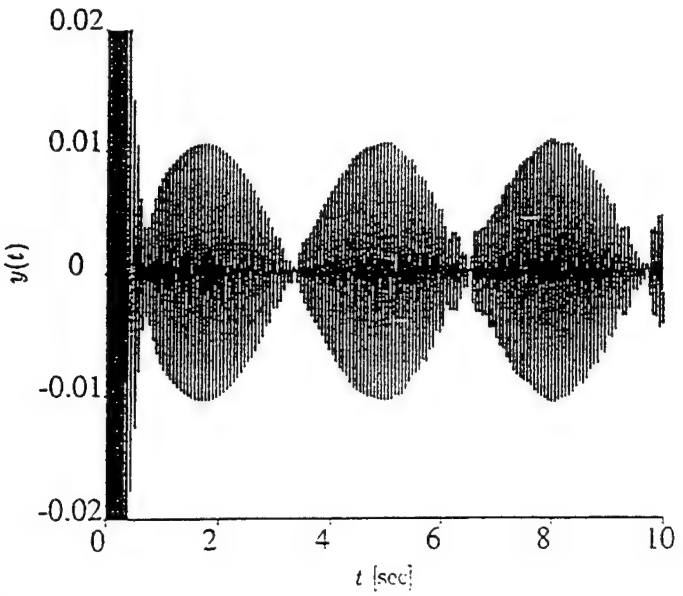

Figure 8: Plant output – Frequency modulation of low frequency

the same plant as in the first simulation, and make the approximation that the output of the plant corresponding to $d(t)$ is the output to $\cos(100t)$, itself modulated by $1 + 0.1\cos(t)$. Then, the additional disturbance can be represented by an equivalent output disturbance with the following components: $\delta n_1(t) = -0.025\cos(t)$, $\delta n_2(t) = 0.025\cos(t)$. It turns out that

$$G^{-1}\begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0 \end{bmatrix}\cos(t), \tag{31}$$

which, because of the diagonal form of the transfer function in (27), implies that only the parameter θ_1 will be affected. This result is expected since only the magnitude of the disturbance changes with time. The analysis (equation (27)) predicts that the parameter variation $\delta\theta_1$ is given by $\delta\theta_1 = 0.099\cos(t) + 0.0099\sin(t)$. Equation (28) gives

$$\begin{bmatrix} \delta\bar{y}_1 \\ \delta\bar{y}_2 \end{bmatrix} = \begin{bmatrix} -0.0002\cos(t) + 0.0025\sin(t) \\ 0.0002\cos(t) - 0.0025\sin(t) \end{bmatrix}. \tag{32}$$

Assuming that $\delta\bar{y} = A(t)\cos(100t) - B(t)\sin(100t)$ where $A(t)$ and $B(t)$ are slowly-varying functions of time, one has that $\delta\bar{y}_1 = A(t)/2$ and $\delta\bar{y}_2 = B(t)/2$, so that

$$\sqrt{A(t)^2 + B(t)^2} = 2\sqrt{\delta\bar{y}_1^2 + \delta\bar{y}_2^2}. \tag{33}$$

This formula leads to a predicted peak magnitude of $\delta\bar{y}$ equal to 0.0070. Note that, in the case where δn models a plant disturbance, $\delta\bar{y}$ is the actual plant output, as δy does not have a physical reality.

Fig. 6 shows $\delta\bar{y}$, the plant output. As expected, the signal appears as a sinusoidal function of high frequency, modulated by a lower frequency signal. The magnitude corresponds to the prediction of the analysis (0.0070). Fig. 7 shows the adaptive parameters. As expected, only the parameter θ_1 which corresponds to the magnitude of the disturbance is affected. The variation follows closely the shape of the disturbance ($0.1\cos(t)$), as predicted by the analysis. The parameter θ_2, which estimates the frequency of the disturbance, remains constant.

When the rate of variation of the magnitude of the disturbance is ten times faster, *i.e.*, the frequency of the modulating sinusoid is 10 rad/s, the numerical analysis predicts that the adaptive parameter satisfies $\delta\theta_1 = 0.05\cos(10t) + 0.05\sin(10t)$. Because of the much higher frequency, the response is delayed and smaller than required for the cancellation of the disturbance. Simulation results validate the analysis for the parameter variations, but are omitted for brevity. In this case, the peak magnitude of the output of the plant was calculated and verified in simulations to be 0.05, which is seven times higher than previously. Note that 10 rad/s is a fairly high frequency for this system, since 10 rad/s is the approximate bandwidth of the closed-loop system.

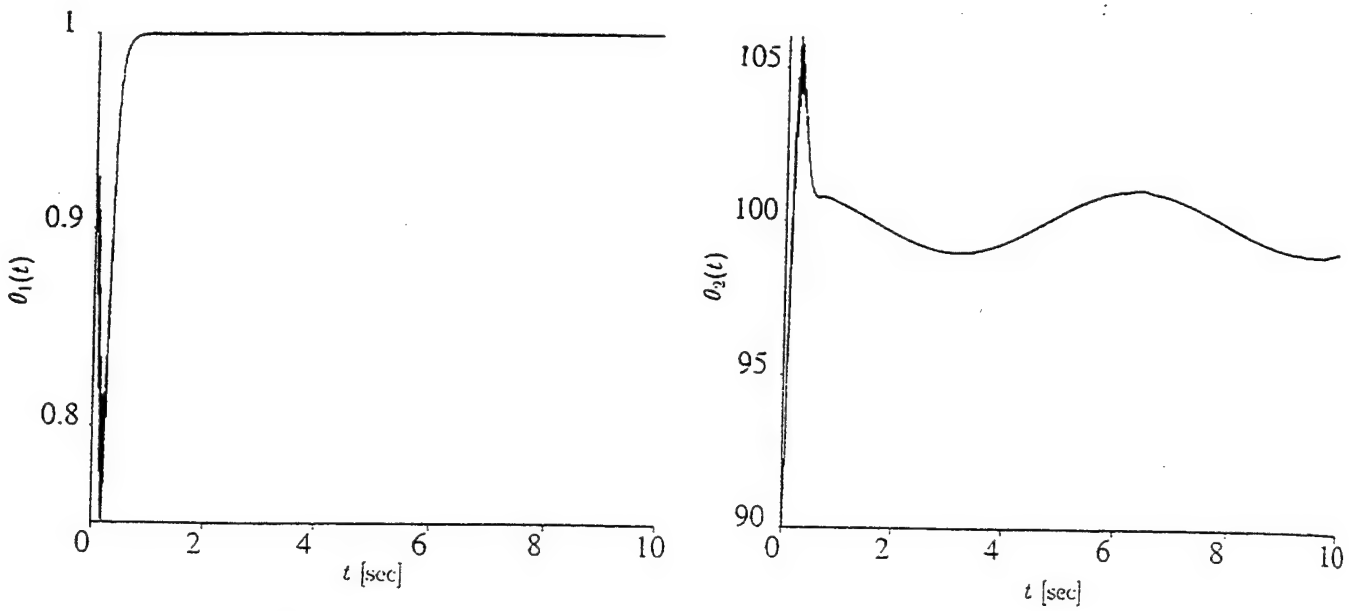

Figure 9: Adaptive parameters – Frequency modulation of low frequency

Next, we consider a similar example but with frequency modulation of the disturbance, so that $d(t) = \cos(100t + \sin(t))$. The instantaneous frequency of this signal is $100 + \cos(t)$. By taking the narrowband approximation of this signal, $d(t) = \cos(100t) - \sin(t)\sin(100t)$. Assuming that the output is equal to the output for the sinusoids at 100 rad/s multiplied by the same slowly-varying coefficients as the input signal, the equivalent output disturbance is calculated as $\delta n_1(t) = -0.25\sin(t)$ and $\delta n_2(t) = -0.25\sin(t)$, with

$$G^{-1}\begin{bmatrix} \delta n_1 \\ \delta n_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \sin(t). \tag{34}$$

The numerical analysis now predicts that $\delta\theta_1 = 0$, which is consistent with the fact that only the instantaneous frequency of the disturbance signal varies. Furthermore, $\delta\theta_2 = 1.015\cos(t) + 0.02\sin(t)$. This function is close to the deviation in the instantaneous frequency of the disturbance signal, which is equal to $\cos(t)$. The predicted peak output magnitude is calculated to be 0.011.

Fig. 8 shows $\delta\bar{y}$, which is the output of the plant. The signal appears as a sinusoidal function of high frequency, modulated by a lower frequency signal whose magnitude corresponds to the prediction of the analysis (0.011). Fig. 9 shows the adaptive parameters. As expected, only the parameter θ_2 which corresponds to the frequency of the disturbance is affected. The variation follows that of the deviation in frequency of the disturbance ($\cos(t)$). The parameter θ_1 which corresponds to the magnitude of the disturbance remains constant.

The case where the instantaneous frequency of the disturbance was $100 + \cos(10t)$ was also considered. Simulation results are omitted for brevity. The analytical result was $\delta\theta_2 = \cos(10t) + \sin(10t)$. In this case, the magnitude of the variation of the parameter is larger than the deviation of the actual parameter, and it leads the actual deviation (in the amplitude modulation case, the deviation was smaller and was lagging). The peak output was calculated and verified in simulations to be 0.0707, which is 7 times higher than the equivalent case for the low frequency. As expected, there is a degradation of performance when the rate of variation of the magnitude or frequency of the disturbance signal is close to the bandwidth of the system.

Finally, note that, in the simulations, G was assumed to be set to the nominal value corresponding to 100 rad/s. Simulations with the nominal G replaced by one that was continuously adjusted using the estimated frequency did not exhibit significant differences in the responses.

6. Conclusions

In this paper, we discussed a method for the rejection of sinusoidal disturbances with unknown frequency. While disturbance rejection for signals with known frequency has been addressed extensively in the literature using various approaches, the analogous task for signals with unknown frequency has received little attention. Such problems are

expected to gain importance, however, in view of the extensive applications where such methods are needed, such as in active noise and vibration control.

The adaptive algorithm incorporated a phase-locked loop concept within a disturbance cancellation control law. The scheme exhibited good convergence properties, and allowed for tracking of signals with slowly varying characteristics. The analysis allowed to predict precisely the loss of performance in the presence of noise and additional disturbances (in terms of residual output error), and the response of the parameters. An area of current research is the extension of the algorithm and of its analysis to the case where multiple frequency components are present, especially when they are harmonically related.

References

[1] U. Emborg and C.F. Ross, "Active Control in the Saab 340," *Proc. Second Conf. Recent Adv. Active Contr. Sound Vib.*, Supplement, pp. S67-S73, Blacksburg, VA, April 1993.

[2] C.A. Shifrin, B. Hill, and C. Donington, "Saab 340Bs Get Active Antinoise System," *Aviation Week and Space Technology*, May 9, 1994, pp. 55-58.

[3] L.J. Eriksson, "A Practical System for Active Attenuation in Ducts," *Sound and Vibration*, vol. 22, no. 2, 1988, pp. 30-34.

[4] J. Simonich, P. Lavrich, T. Sofrin, and D. Topol, "Active Aerodynamic Control of Wake-Airfoil Interaction Noise - Experiment," *AIAA Journal*, vol. 31, no. 10, Oct. 1993, pp. 1761-1768.

[5] E. Phillips, "Tests Show Active Flaps Reduce Helicopter Noise," *Aviation Week and Space Technology*, April 18, 1994, pp. 38-41.

[6] S.R. Hall and N.M. Wereley, "Performance of Higher Harmonic Control Algorithms for Helicopter Vibration Reduction," *J. Guidance, Control and Dynamics*, vol. 16, no. 4, 1993, pp. 793-797.

[7] M. Bodson, A. Sacks and P. Khosla, "Harmonic Generation in Adaptive Feedforward Cancellation Schemes," *IEEE Trans. on Automatic Control*, vol. 39, no. 9, 1994, pp. 1939-1944.

[8] A. Sacks, M. Bodson, and P. Khosla, "Experimental Results of Adaptive Periodic Disturbance Cancellation in a High Performance Magnetic Disk Drive," *Proc. of the American Control Conference*, San Francisco, CA, 1993, pp. 686-690. To appear in *ASME Journal on Dynamics and Control.*

[9] W. Messner and M. Bodson, "Design of Adaptive Feedforward Algorithms Using Internal Model Equivalence," *International Journal of Adaptive Control and Signal Processing*, vol. 9, 1995, pp. 199-212.

[10] A. Sacks, M. Bodson, and W. Messner, "Advanced Methods for Repeatable Runout Compensation," *IEEE Trans. on Magnetics*, vol. 31, no. 2, 1995, pp. 1031-1036.

[11] J.R. Glover, "Adaptive Noise Canceling Applied to Sinusoidal Interferences," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 6, 1977, pp. 484-491.

[12] M. Feder, "Parameter Estimation and Extraction of Helicopter Signals Observed with a Wide-Band Interference," *IEEE Trans. on Signal Processing*, vol. 41, no. 1, 1993, pp. 232-244.

[13] P.A. Regalia, "An Improved Lattice-Based Adaptive IIR Notch Filter," *IEEE Trans. on Signal Processing*, vol 39, no. 9, 1991, pp. 2124-2128.

[14] M. Bodson and S. Douglas, "Adaptive Algorithms for the Rejection of Periodic Disturbances with Unknown Frequency," to appear in *Proc. 1996 IFAC Congress*, San Francisco, CA, June 1996.

[15] A.R. Hambley, *An Introduction to Communication Systems*, W.H. Freeman & Company, New York, 1990.

[16] D.H. Wolaver, *Phase-Locked Loop Circuit Design*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

Pergamon

PII: S0005-1098(97)00149-0

Automatica, Vol. 33, No. 12, pp. 2213–2221, 1997

Printed in Great Britain
0005-1098/97 $17.00 + 0.00

Brief Paper

Adaptive Algorithms for the Rejection of Sinusoidal Disturbances with Unknown Frequency*

MARC BODSON† and SCOTT C. DOUGLAS†

Key Words—Adaptive algorithms; active noise control; sinusoidal signals; disturbance rejection; phase-locked loops.

Abstract—Two algorithms are presented for the rejection of sinusoidal disturbances with unknown frequency. The first is an indirect algorithm where the frequency of the disturbance is estimated, and the estimate is used in another adaptive algorithm that adjusts the magnitude and phase of the input needed to cancel the effect of the disturbance. A direct algorithm that uses the concept of a phase-locked loop is also presented in which frequency estimation and disturbance cancellation are performed simultaneously. Approximate analyses are presented for both schemes and the results are found useful for the selection of the design parameters. Simulations are given which demonstrate the validity of the analytical results and the ability of the algorithms to reject sinusoidal disturbances with unknown frequency. The indirect algorithm is found to have a larger capture region for the parameter estimates, whereas the direct algorithm has superior convergence properties locally about the optimum parameter estimates © 1997 Elsevier Science Ltd. All rights reserved.

1. INTRODUCTION

In this paper, we consider the problem of attenuating an output signal $y(t)$ that, in the Laplace domain, is given by

$$y(s) = P(s)(u(s) - d(s)), \tag{1}$$

where $u(s)$ and $d(s)$ are the Laplace transforms of the controller output and disturbance signals, respectively, and $P(s)$ is the Laplace transform of the impulse response of the plant. The plant is assumed to be linear, time-invariant, and stable. The goal of the control system is to generate $u(t)$ such that $y(t) \to 0$ as $t \to \infty$. We assume that the disturbance $d(t)$ is a sinusoid of fixed magnitude θ_1^* and fixed frequency ω_1 such that

$$d(t) = \theta_1^* \cos(\alpha_d(t)), \qquad \dot{\alpha}_d(t) = \omega_1. \tag{2}$$

In this task, the parameters θ_1^*, ω_1, and $\alpha_d(0)$ are all unknown.

Algorithms for solving the above problem have use in a wide range of applications. Of particular interest is the problem of active noise and vibration control, in which $d(t)$ is an offending noise or vibration source and $P(s)$ is the transfer function of the output-actuator-to-error-sensor propagation path. Often, the noise source consists mainly of periodic components due to rotating machinery generating the undesired noise signal. Examples of such noises include engine noise in turboprop aircraft (Emborg and Ross, 1993), engine noise in automobiles (Shoureshi and Knurek, 1996), and ventilation noise in HVAC systems (Eriksson, 1988). In practice, the frequency of the disturbance is usually not known and may even vary during operation. In these cases, it is desirable to place an encoder or tachometer on the rotating machine that is at the origin of the disturbance to measure the frequency of the disturbance. Alternatively, if a sensor can be placed near the source of the disturbance such that $d(t)$ can be accurately characterized, then the control task reduces to that of adaptive feedforward control (AFC) (see Nelson and Elliott, 1992). However, it is impossible to use such a sensor in applications where the addition of the sensor is too costly or reduces the reliability of the overall system due to strong vibrations, high temperatures, or dirty conditions within the environment.

In this paper, we present two adaptive control algorithms for the rejection of sinusoidal disturbances with unknown frequency. These systems only require a single sensor located at the output of the plant for their operation. The first approach combines an AFC scheme for attenuating sinusoidal disturbances with known frequency as described and analyzed in Bodson *et al.* (1994) and Sacks *et al.* (1996), together with an algorithm to estimate the frequency of the disturbance. This approach is called *indirect* because the frequency of the disturbance is estimated independently of the cancellation scheme. The second approach consists

*Received 12 September 1995; revised 27 December 1996; received in final form 13 June 1997. This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.
Corresponding author Dr Marc Bodson. Tel. (801) 581 8590; Fax (801) 581 5281; E-mail bodson@ee.utah.edu.

†Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112, U.S.A.

in extending the AFC scheme of Bodson *et al.* (1994) and Sacks *et al.* (1996) by integrating a *phase-locked loop* within the scheme so that disturbances with unknown frequency can be directly cancelled. Our analyses of these two methods enable the design of the systems to provide useful rejection of sinusoidal disturbances, and simulations validate the theoretical results. Our results indicate that the direct algorithm has superior convergence properties locally about the optimum convergent point of the controller, whereas the indirect algorithm provides a wider capture region for the system's adaptive parameters.

2. INDIRECT APPROACH

2.1. *Frequency estimation*

Our development of the indirect algorithm relies on a method for the estimation of the frequency of a signal. Here, we consider an adaptive notch filter developed in Regalia (1991) that is designed to eliminate one (or more) periodic component(s) from a measured signal. This adaptive notch filter estimates the frequency of the unknown signal as part of its operation. In our work, we transpose Regalia's algorithm to continuous-time so that standard averaging methods can be used to analyze the system's equations. It is relatively straightforward to transfer the results of our control system design back to the discrete-time domain for implementation purposes.

Consider the problem of estimating the frequency ω of a signal $y(t)$, where θ_f is the resulting estimate. The continuous-time version of Regalia's algorithm has three states θ_f, x_1, and x_2, that satisfy the differential equations

$$\dot{x}_1 = x_2, \tag{3}$$

$$\dot{x}_2 = -2\zeta\theta_f x_2 - \theta_f^2 x_1 + ky, \tag{4}$$

$$\dot{\theta}_f = -g_1(ky - 2\zeta\theta_f x_2)x_1. \tag{5}$$

Note that this system of equations is highly nonlinear. The three parameters of the system are the adaptation gain g_1, the damping factor ζ, and a filter gain k, all of which are positive-valued. The algorithm's behavior can be explained through an averaging analysis, justified for small values of g_1 and for a periodic signal $y(t)$ (cf. Sastry and Bodson, 1989). For θ_f constant,

$$x_1 = \frac{k}{D(s,\theta_f)}[y], \qquad x_2 = \frac{ks}{D(s,\theta_f)}[y], \tag{6}$$

where $D(s,\theta_f) = s^2 + 2\zeta\theta_f s + \theta_f^2$. Furthermore,

$$ky - 2\zeta\theta_f x_2 = \frac{k(s^2+\theta_f^2)}{D(s,\theta_f)}[y]. \tag{7}$$

By application of averaging theory, the evolution of the state θ_f can be approximated by the solution of the averaged system, which is given by

$$\dot{\theta}_{av} = -g_1 AVG\left[\frac{k(s^2+\theta_{av}^2)}{D(s,\theta_{av})}[y]\cdot\frac{k}{D(s,\theta_{av})}[y]\right]. \tag{8}$$

Assuming that $y(t) = \sum_{i=1}^{n} r_i \sin(\omega_i t + \phi_i)$, the averaged system is given by

$$\dot{\theta}_{av} = -\frac{g_1}{2}\sum_{i=1}^{n}\frac{k^2 r_i^2(\theta_{av}^2 - \omega_i^2)}{(\theta_{av}^2-\omega_i^2)^2 + (2\zeta\theta_{av}\omega_i)^2}. \tag{9}$$

If the signal $y(t)$ has a single sinusoidal component, the analysis indicates that the averaged system has a pair of equilibrium points at $\theta_{av} = \pm\omega_1$. Around ω_1, the linearized system is exponentially stable, with dynamics

$$\dot{\theta}_{av} = -\frac{g_1 k^2 r_1^2}{4\zeta^2\omega_1^3}(\theta_{av} - \omega_1). \tag{10}$$

In other words, the algorithm is able to identify the frequency of the signal ω_1. Even in the presence of multiple sinusoids, the algorithm is able to lock onto a main frequency component despite the presence of competing tones. Typically, the averaged system will exhibit multiple equilibrium points. If the parameter ζ is small, the first term in the sum of (9) will dominate the others when θ_{av} is close to ω_1. Therefore, in the neighborhood of $\theta_{av} = \omega_1$, the right-hand side of (9) will be approximately the same as that of the system with only one sinusoidal component. The same argument can be repeated for the other frequencies, so that there will be an equilibrium point associated with each of the sinusoidal components of the signal $y(t)$. The initial value of θ_f and the magnitudes of the frequency components r_i affect the frequency towards which θ_f will converge.

2.2. *Indirect adaptive algorithm*

The algorithm for frequency estimation can be combined with the adaptive feedforward control scheme in Sacks *et al.* (1996) to produce an algorithm capable of attenuating periodic disturbances with unknown frequency. The control law is given by

$$u(t) = \theta_c(t)\cos(\alpha(t)) - \theta_s(t)\sin(\alpha(t)),$$
$$\dot{\alpha}(t) = \theta_f(t), \tag{11}$$

where the two adaptive parameters θ_c and θ_s are updated as

$$\frac{d}{dt}\begin{bmatrix}\theta_c\\ \theta_s\end{bmatrix} = -g_2 G^{-1}\begin{bmatrix} y\cos(\alpha)\\ -y\sin(\alpha)\end{bmatrix}, \tag{12}$$

and

$$G = \frac{1}{2}\begin{bmatrix}P_R & -P_I\\ P_I & P_R\end{bmatrix}; \qquad P_R = \mathrm{Re}[P(j\omega_1)];$$
$$P_I = \mathrm{Im}[P(j\omega_1)]. \tag{13}$$

The parameter $g_2 > 0$ is an arbitrary adaptation gain. The averaging analysis of (12) and (13) shows that, for small values of the gain g_2, the dynamics of this adaptive system are approximately the same as those of two decoupled, first-order systems, with their poles located at $-g_2$(rad/s).

Note that the algorithm in (12) and (13) uses the value of the frequency response of the plant at the frequency of the disturbance ω_1. However, the stability of (12) for small gain g_2 is guaranteed so long as the phase of the frequency response is correct to within $\pm 90°$. Therefore, a rough estimate of ω_1 in (12) will be adequate in most cases. If desired, ω_1 may also be replaced by θ_f in (13).

2.3. *Simulations of the indirect algorithm*

We now present simulations of the indirect algorithm for a plant given by $P(s) = 100/(s + 100)$ and a sinusoidal disturbance $d(t) = \cos(100t)$. In the frequency estimation algorithm, we choose the initial states to be equal to zero, and we choose $\zeta = 0.1$, $k = 100$, and $g_1 = 1000$. With these parameter choices, the pole of the linearized system in (10) is at -125 rad/s. In the AFC scheme, the initial states of the system are set to zero, and we choose $g_2 = 10$. Such a parameter choice yields an averaged system with two real poles at -10 rad/s, as predicted by the analysis in Sacks *et al.*, (1996).

For our first simulation, frequency estimation and disturbance cancellation are performed separately. Specifically, for the first second of the simulation, the parameters of the AFC scheme are frozen while an accurate estimate of the frequency of the disturbance is calculated. Then, the frequency estimate is fixed for $t \geq 1$ and is used by the AFC scheme to adjust the amplitudes of the $\cos(\cdot)$ and $\sin(\cdot)$ components of the controller to attenuate the disturbance.

Figure 1 shows the frequency estimate produced by the algorithm, as indicated by the solid line in the figure. For our parameter choices, the frequency estimation algorithm in (12) and (13) converges within the first second of the simulation to the value $\theta_f = 99.95$ rad/s, a value close to $\omega_1 = 100$ rad/s. The response predicted by the nonlinear averaged system in (8) is shown as a dashed line on this figure and is seen to closely approximate the true system's transient response. Note that the frequency estimation algorithm converges to its proper setting despite a 100% initial error in the frequency estimate, and the ability of this algorithm to lock onto the frequency of the unknown disturbance is useful in many practical situations.

Figure 2 shows the logarithm of the output of the plant, computed as $\log(|y(t)| + \varepsilon)$ where $\varepsilon = 10^{-6}$. The envelope of the plant output is constant during the frequency estimation phase of the controller's operation and is seen to decrease rapidly once the AFC algorithm is engaged. The system's output does not decrease significantly after a period of time. The evolutions of θ_c and θ_s are shown in Fig. 3 as a solid line and a dashed line, respectively. While the parameters appear to be drifting for $t > 2$ s, their variation is in fact sinusoidal at a frequency equal to the difference between the true and estimated frequencies of disturbance, and the sum of

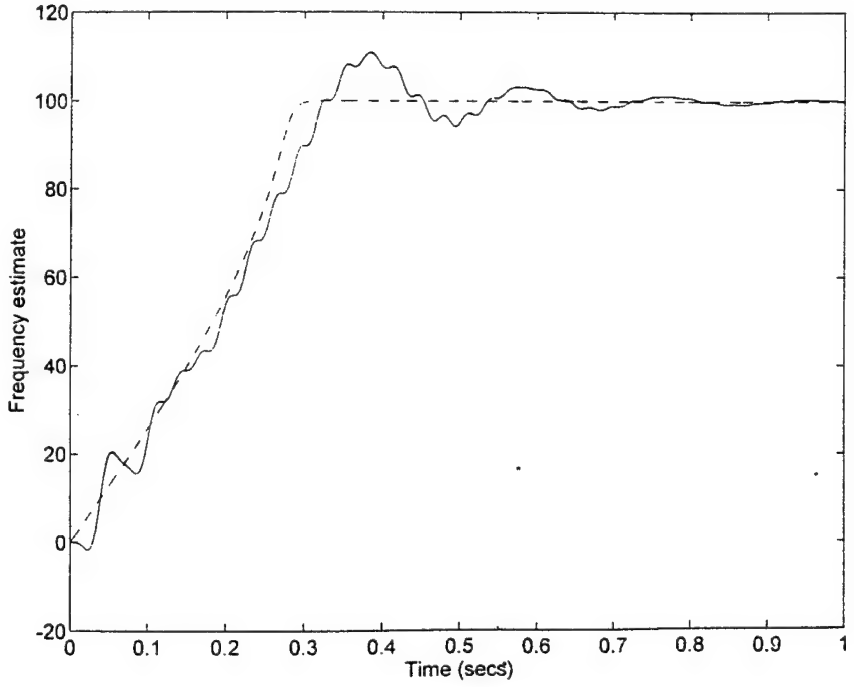

Fig. 1. Frequency estimate for the indirect algorithm—separate adaptation.

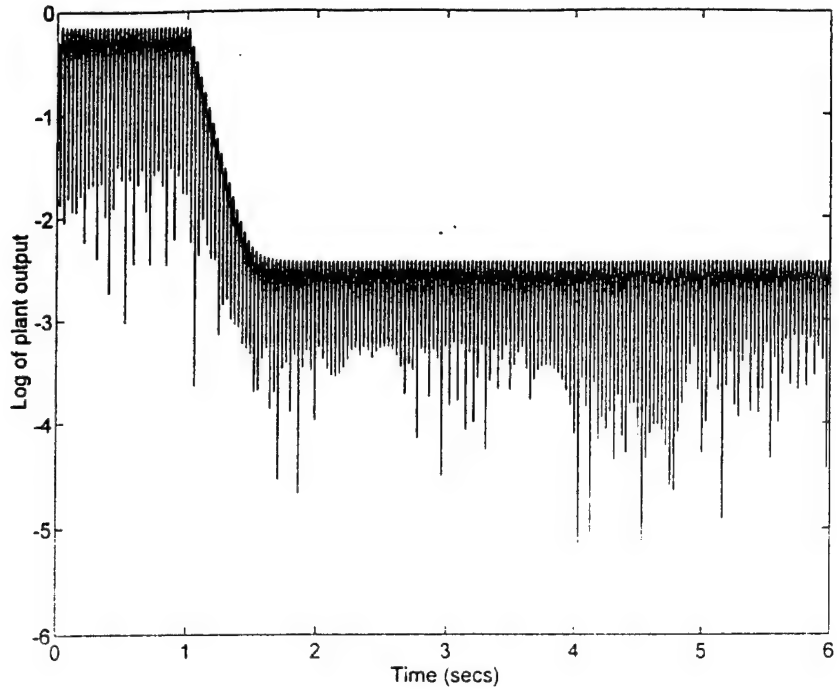

Fig. 2. Log of the plant output for the indirect algorithm—separate adaptation.

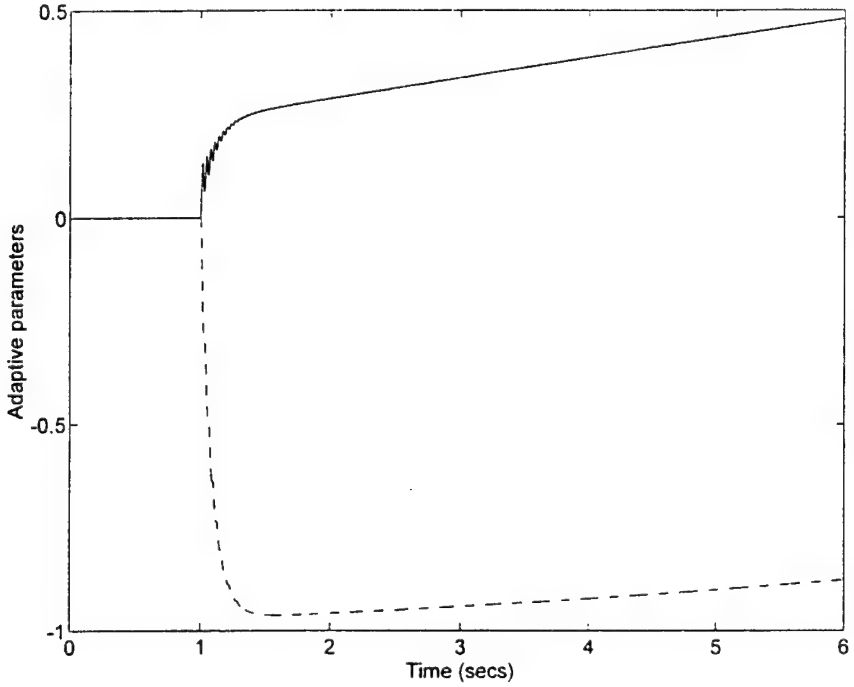

Fig. 3. Adaptive parameters for the indirect algorithm—separate adaptation.

the squares of the parameters remains close to one over this time period. Note that continuous variations of the amplitudes of the $\cos(\cdot)$ and $\sin(\cdot)$ components of the controller output are necessary to reduce the disturbance amplitude when the frequency estimate θ_f is not exact, resulting in a low residual output error in steady state.

We now explore the performance of the indirect scheme in a true adaptive mode in which all the parameters are adjusted simultaneously. Although this implementation might be expected to resolve the convergence problems observed with separate parameter adaptation stages, such is not the case. The log of the plant output is shown in Fig. 4.

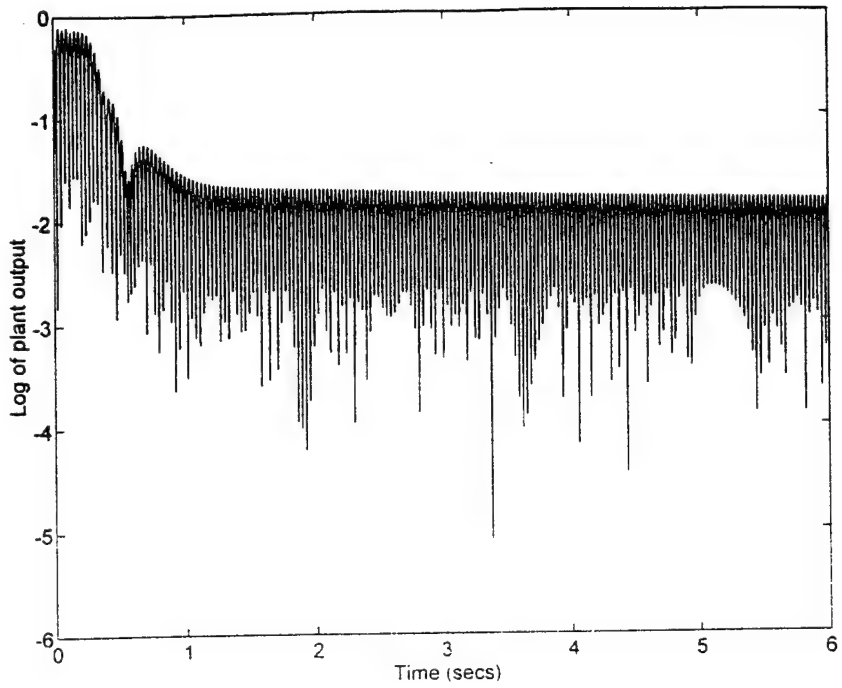

Fig. 4. Log of the plant output for the indirect algorithm simultaneous adaptation.

Although this implementation yields a more rapid decrease of the plant output, a substantial residual error remains at the end of the simulation run. In this case, the lack of asymptotic convergence may be attributed to the fact that the control input eliminates the signal which is used for the frequency estimation, thus preventing accurate convergence of θ_f. This problem is an indication of the fact that the indirect scheme is overparameterized and can be resolved by considering a direct adaptive scheme in which the amplitude and phase of the sinusoidal component are computed via a single combined procedure, as we now show.

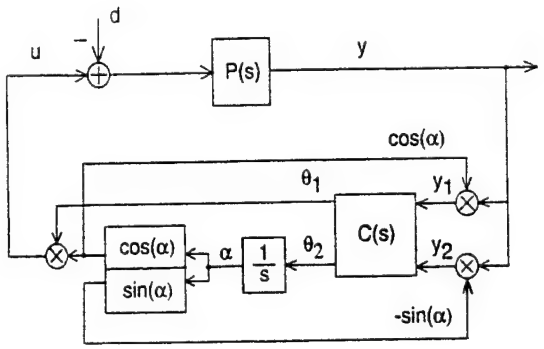

Fig. 5. Direct adaptive algorithm for sinusoidal disturbance cancellation.

3. A DIRECT ALGORITHM

3.1. *Adaptive algorithm*

An alternative approach to the indirect scheme of the previous section is a direct scheme in which a single error signal is used to update the frequency and the magnitude estimates simultaneously. Here, one such algorithm is presented that combines elements of the AFC scheme discussed earlier with a modified version of a phase-locked loop structure commonly used in communication systems (Hambley, 1990).

This scheme is shown in Fig. 5. In the figure, θ_1 is the estimate of the magnitude of the disturbance signal, and θ_2 is the estimate of its instantaneous frequency. Moreover, α is the estimate of the phase of the disturbance signal and is the integral of the estimate of the frequency θ_2. The equations for the control algorithm are

$$u = \theta_1 \cos(\alpha), \quad \dot{\alpha} = \theta_2, \quad y_1 = y\cos(\alpha), \quad y_2 = -y\sin(\alpha). \tag{14}$$

The user-defined transfer function matrix $C(s)$ relates the signals y_1 and y_2 to the parameters θ_1 and θ_2, respectively. An approximate analysis of this adaptive system will allow us to deduce a simple procedure for the design of $C(s)$ to obtain adequate performance from this system.

3.2. *Approximate analysis*

Our analysis is based on a fundamental fact, whose proof is reminiscent of derivations found in the analysis of frequency-modulation communication systems (cf. Hambley, 1990).

Assumptions.

- The values of θ_1 and θ_2 vary sufficiently slowly that the response of the plant to the signal $u(t)$ can be approximated by the steady-state output of the plant for a sinusoidal input with frequency θ_2.
- The instantaneous frequency θ_2 is close to ω_1, such that $P(j\theta_2)$ can be replaced by $P(j\omega_1)$.

Basic fact. Considering low-frequency components only, the two signals $y_1(t)$ and $y_2(t)$ are approximately given by

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = G \begin{bmatrix} \theta_1(t) - d_1 \cos(\alpha(t) - \alpha_d(t)) \\ d_1 \sin(\alpha(t) - \alpha_d(t)) \end{bmatrix}, \quad (15)$$

where G is as defined in (13).

Proof. Under the assumptions, the output of the plant is given by

$$\begin{aligned} y(t) = {} & P_R\theta_1(t)\cos(\alpha(t)) - P_I\theta_1(t)\sin(\alpha(t)) \\ & - P_R d_1 \cos(\alpha_d(t)) + P_I d_1 \sin(\alpha_d(t)). \end{aligned} \quad (16)$$

Keeping only the low-frequency components of the signals y_1 and y_2, we find that

$$\begin{aligned} y_1(t) = {} & \tfrac{1}{2}P_R\theta_1(t) - \tfrac{1}{2}P_R d_1 \cos(\alpha(t) - \alpha_d(t)) \\ & - \tfrac{1}{2}P_I d_1 \sin(\alpha(t) - \alpha_d(t)), \\ y_2(t) = {} & \tfrac{1}{2}P_I\theta_1(t) + \tfrac{1}{2}P_R d_1 \sin(\alpha(t) - \alpha_d(t)) \\ & - \tfrac{1}{2}P_I d_1 \cos(\alpha(t) - \alpha_d(t)), \end{aligned} \quad (17)$$

and the result is obtained.

Comments. The elimination of the high-frequency components within the system can be achieved by proper low-pass filtering of the signals $y_1(t)$ and $y_2(t)$. In the algorithm discussed in this paper, the signals are applied to a compensator $C(s)$ which is low-pass in nature. Although the filtering is far from ideal, simulations show that our approximations are satisfactory for the compensator design without the need for additional filtering.

3.3. *Compensator design*

Equation (15) can be viewed as an alternative description of the plant, with two inputs θ_1 and α and two outputs y_1 and y_2. Although this equation is nonlinear, a linear system is obtained if the phase error $\alpha - \alpha_d$ is small. This system is described by

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = G \begin{bmatrix} \theta_1(t) - d_1 \\ d_1(\alpha(t) - \alpha_d(t)) \end{bmatrix}. \quad (18)$$

Using this linearized result, several methods for designing $C(s)$ can be used. Our approach, while not the best or the most sophisticated, yields a simple design for implementation purposes. We define two variables x_1 and x_2 as

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = G^{-1} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \quad (19)$$

so that

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \theta_1(t) - d_1 \\ d_1\left(\int_0^t (\theta_2(\sigma) - \omega_1)d\sigma + \alpha(0) - \alpha_d(0)\right) \end{bmatrix}. \quad (20)$$

As a result, the dynamics of the system from the parameters θ_1 and θ_2 to the variables x_1 and x_2 are decoupled from one another and are those of a gain of 1 and of an integrator with a gain d_1, respectively. The unknown parameters d_1, ω_1, $\alpha_d(0)$ act as constant disturbances. The compensator $C(s)$ may then be designed as the cascade of the transformation (19) and control laws of the form

$$\theta_1 = \frac{C_1(s)}{s}[x_1], \qquad \theta_2 = \frac{C_2(s)}{s}[x_2]. \quad (21)$$

where the integrators are included to reject the disturbances composed of d_1 and $\alpha_d(0)$. The transfer functions $C_1(s)$ and $C_2(s)$ are designed to guarantee the closed-loop stability of the two systems $P_1(s) = 1$ and $P_2(s) = d_1/s$. One possible choice is

$$C_1(s) = -g_1, \qquad C_2(s) = -g_2\frac{s+a}{s+b}. \quad (22)$$

Because the magnitude of the disturbance d_1 acts as a gain in the second transfer function, the parameters of the compensator $C_2(s)$ must be designed for a range of magnitudes of the disturbance d_1. Simple techniques can also be used to estimate the disturbance level before the system is turned on.

The algorithm requires the knowledge of the matrix G, which depends on ω_1. Similar to the indirect algorithm, one can resolve this problem in two ways. One may choose to design a compensator that works satisfactorily for a range of matrices G corresponding to a range of frequencies of the unknown disturbance. Often, the uncertainty in the frequency of the disturbance is small enough such that variations in the G matrix are inconsequential, yielding adequate performance for a compensator designed for a mid-range disturbance frequency. Alternatively, one could use for G the equivalent transfer function matrix corresponding to the estimated frequency $\dot{\alpha} = \theta_2$. In either case, knowledge of the frequency response of the plant in the frequency range of interest is required.

3.4. *Simulations of the direct algorithm*

We now explore the performance of the direct algorithm via simulation. We consider the same situation as in the indirect algorithm, in which

$P(s) = 100/(s + 100)$, $\omega_1 = 100$, and $d_1 = 1$. For the direct algorithm, the parameter g_1 is set to 10, leading to a closed-loop pole for the first system at -10 rad/s. The other parameters are set to $g_2 = 400$, $a = 5$, and $b = 30$, leading to three closed-loop poles for the second system located at -10 rad/s and $-10 \pm j10$ rad/s. The initial states of all parameters are zero except for $\theta_1(0) = 0.9$, $\theta_2(0) = 90$ rad/s, and $\alpha(0) = 90°$. Note that, since $\alpha_d(0) = 0$, the system is initialized with a large phase error, such that the linearized analysis is less accurate.

Figure 6 shows the log of the output of the plant. The output is found to decrease to negligible values in less than two seconds. The transient behavior of the magnitude estimate θ_1 is shown in Fig. 7, where

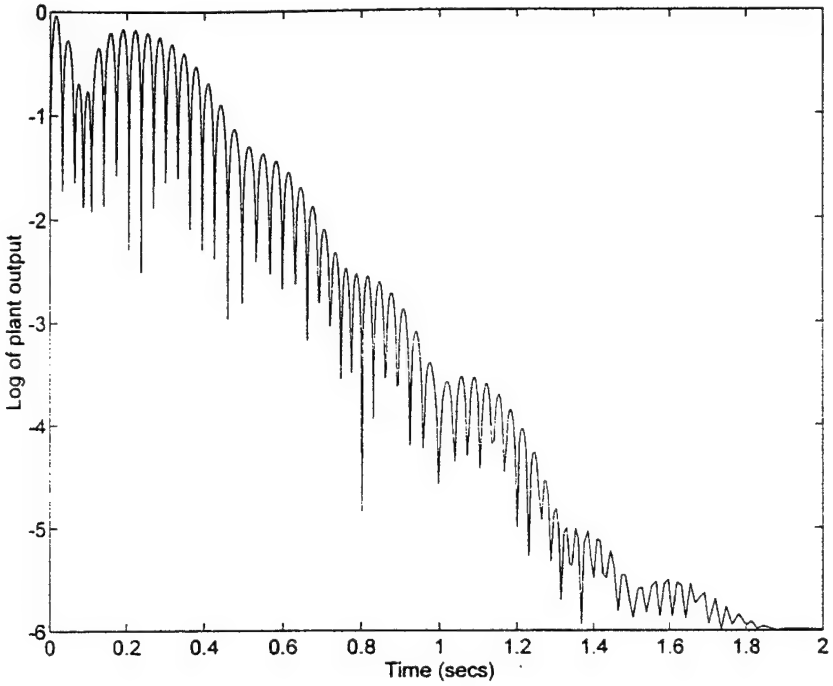

Fig. 6. Log of the plant output for the direct algorithm.

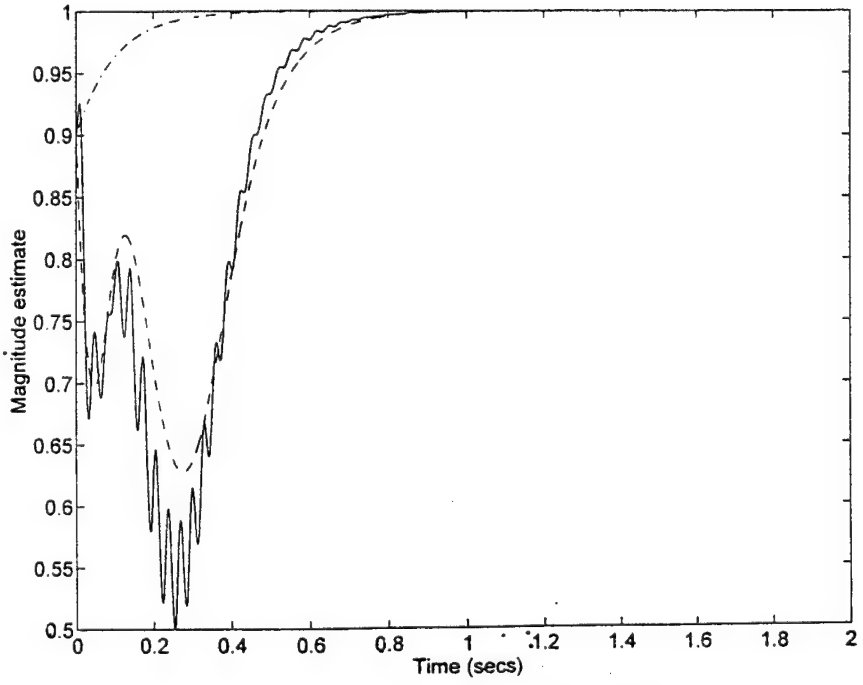

Fig. 7. Magnitude estimate for the direct algorithm.

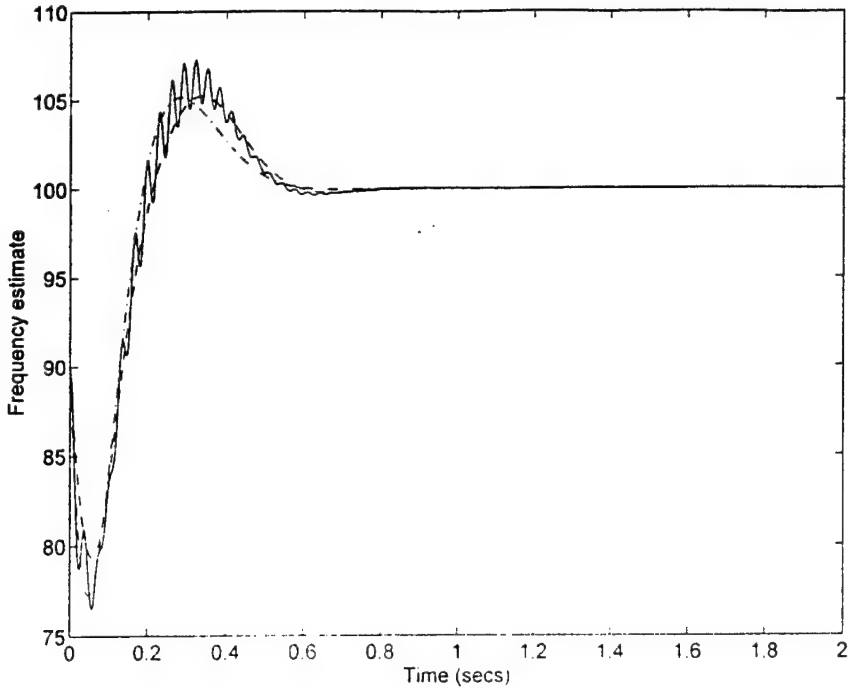

Fig. 8. Frequency estimate for the direct algorithm.

the solid line is the parameter response, the dashed line is the response predicted using the nonlinear approximation in (15), and the dot-dashed line is the response predicted using the linear approximation in (18). The equivalent behaviors of the actual and theoretically-predicted values of the frequency estimate θ_2 are shown in Fig. 8. Note that the matches between the theoretical and actual behaviors of the estimates are particularly good when using the nonlinear approximation, whereas the analysis using the linear approximation is only accurate for the frequency estimate. Although the nonlinear effects are clearly significant, our design based on the linear approximation provides adequate convergence of the system.

In these simulations, the system converged despite a 10% initial error in frequency. It was found that the scheme was able to acquire frequencies with errors up to 30% in this case. It is well-known that an important characteristic of phase-locked loops is their *lock-in* (or *pull-in*) range (Hambley, 1990). Careful design of the loop transfer function can increase the lock-in range and improve performance. Additional supervisory logic might also be used for acquisition, and would be part of a practical design. In addition, although the elements of G were set to their nominal values corresponding to a disturbance frequency of 100 rad/s for these plots, simulations with the nominal G replaced by a continuously-adjusted matrix according to the estimated frequency of the disturbance did not exhibit significant differences in transient behaviors. Other simulations with slowly varying disturbances and with measurement noises indicate that the direct algorithm performs well in these situations, and the effects of these variations can be analyzed precisely using the linear approximation, as described in Bodson and Douglas, (1996) and Bodson, (1996).

4. CONCLUSIONS

In this paper, we have presented two methods for the rejection of sinusoidal disturbances with unknown frequency. While disturbance rejection for signals with known frequency has been addressed extensively in the literature using various approaches, the analogous task for signals with unknown frequency has received comparatively little attention despite its importance in applications such as active noise and vibration control. Our study of an algorithm for the determination of the frequency of an unknown signal yielded an averaging analysis that was useful for system design. It was shown that this frequency estimation algorithm could be combined with an AFC algorithm to obtain an indirect algorithm for the cancellation of disturbances with unknown frequency. While this system was able to lock on a sinusoidal signal with no *a priori* information about the frequency of the disturbance, the convergence properties of the scheme were found to be less than ideal. A second algorithm incorporating a phase-locked loop

within the cancellation scheme yielded improved convergence properties, and our averaging analysis provided a precise prediction of the dynamic characteristics of the system. A successful scheme for sinusoidal disturbance cancellation scheme would combine the direct algorithm presented here with an initialization scheme for providing rough initial estimates of the disturbance frequency. Simulations verified the results of the analysis and indicated the usefulness of the algorithms for the sinusoidal disturbance cancellation task.

REFERENCES

Shoureshi, R. and P. Knurek (1996). Automotive applications of a hybrid active noise and vibration control. *IEEE Control Systems*, **16**(6), 72–78.

Bodson, M. (1996). Cancellation of sinusoidal disturbances with unknown frequency. *Proc. 9th Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, pp. 174–179.

Bodson, M. and S. C. Douglas (1996). Adaptive algorithms for the rejection of periodic disturbances with unknown frequency. *Proc. 13th World Congress of the Int. Federation of Automatic Control*, San Francisco, CA, Vol. K, pp. 229–234.

Bodson, M., A. Sacks and P. Khosla (1994). Harmonic generation in adaptive feedforward cancellation schemes. *IEEE Trans. Automat. Control*, **39**(9), 1939–1944.

Emborg, U. and C. F. Ross (1993). Active control in the Saab 340. *Proc. 2nd Conf. Recent Adv. Active Control Sound Vibrations*, (Suppl.) Blacksburg, VA, pp. S67–S73.

Eriksson, L. J. (1988). A Practical system for active attenuation in ducts. *Sound and Vibration*, **22**(2), 30–34.

Hambley, A. R. (1990). *An Introduction to Communication Systems*. Freeman, New York.

Nelson, P. A. and S. J. Elliott (1992). *Active Control of Sound*. Academic Press, London.

Regalia, P. A. (1991). An improved lattice-based adaptive IIR notch filter. *IEEE Trans. Signal Process.*, **39**(9), pp. 2124–2128.

Sacks, A., M. Bodson and P. Khosla (1996). Experimental results of adaptive periodic disturbance cancellation in a high performance magnetic disk drive. *ASME J. Dynamic Systems Measurement Control*, **118**, 416–424.

Sastry, S. and M. Bodson (1989). *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Englewood Cliffs, NJ.

Narrowband Disturbance Rejection using Adaptive Feedback Algorithms

Marc Bodson and Scott C. Douglas *

Department of Electrical Engineering, University of Utah
Salt Lake City, UT 84112

ABSTRACT

Two adaptive algorithms that reject narrowband disturbances of unknown frequency are compared. The first is based on an adaptive implementation of the internal model principle. The second is based on a phase-locked loop structure and generates a signal whose magnitude and phase match those of the disturbance. The adaptive internal model principle algorithm provides global stability under ideal conditions, tracks reference inputs, and can stabilize unstable plants. However, the algorithm suffers from convergence and robustness problems in simulations and does not appear to be well-suited to high-order systems or systems with unknown time delays, two situations that are typically encountered in active noise control applications. The algorithm based on the phase-locked loop concept has better convergence and robustness properties and can be applied to a broad range of stable systems. Some prior information is required about the magnitude and frequency of the disturbance and about the frequency response of the plant, however. A real-time implementation of the phase-locked loop scheme in an active noise control task provides a 25 dB reduction of a sinusoidal tone of unknown frequency.

Keywords: adaptive control, signal processing, periodic disturbances, phase-locked loops, active noise control.

1. INTRODUCTION

We consider the problem of attenuating an output signal $y(t)$ that, in the Laplace domain, is given by

$$y(s) = P(s)(u(s) - d(s)), \tag{1}$$

where $u(s)$ and $d(s)$ are the Laplace transforms of the controller output and disturbance signals, respectively, and $P(s)$ is the Laplace transform of the impulse response of the plant. The plant is assumed to be linear and time-invariant. The goal of the control system is to generate $u(t)$ such that $y(t) \to 0$ as $t \to \infty$. We assume that the disturbance $d(t)$ is a sinusoid of fixed magnitude $\theta_1{}^*$ and fixed frequency ω_1

$$\begin{aligned} d(t) &= \theta_1{}^* \cos(\alpha_d(t)), \\ \dot{\alpha}_d(t) &= \omega_1. \end{aligned} \tag{2}$$

In this task, the parameters $\theta_1{}^*$, ω_1, and $\alpha_d(0)$ are all unknown.

The problem formulation is typical of active noise control problems with a periodic noise source [1]. For simplicity, it is assumed that a single tone is present, although the algorithms of the paper can be extended to multiple tones. Feedback algorithms are considered because a separate measurement of the disturbance may not be available or may be costly to acquire. Such algorithms are more difficult to design than the adaptive feedforward control algorithms that are often used in active noise control tasks.

*Further author information -
M.B.: E-mail: bodson@ee.utah.edu; Telephone: 801-581-8590; Fax: 801-581-5281.
S.C.D.: E-mail: douglas@ee.utah.edu; Telephone: 801-581-4445; Fax: 801-581-5281.

One approach to this problem is an adaptive implementation of the internal model principle of the form proposed in [2], [3], and [4]. The algorithms address a broad class of problems, including those in which the plant is unknown and a reference input must be tracked. In this paper,we compare the respective advantages and disadvantages of these approaches with an alternative approach employing a phase-locked loop within the system [5], [6]. As we shall show, the implementation of the internal model principle algorithms presents some practical difficulties that are largely overcome by the phase-locked loop algorithm.

2. ADAPTIVE INTERNAL MODEL PRINCIPLE

2.1 Algorithm

The internal model principle prescribes that the transfer function of a feedback control system designed to reject a sinusoidal disturbance of frequency $j\omega_1$ should have a pair of poles at $s = \pm j\omega_1$. The infinite loop gain at the frequency ω_1 ensures that the sinusoidal disturbance is perfectly rejected if the closed-loop system is stable. This objective can be realized for disturbances of unknown frequency and for unknown plants with an adaptive implementation of the internal model principle. The algorithms proposed by [2], [3], and [4] are obtained from standard adaptive control theory by choosing an overparameterized compensator. For a plant of order n that would normally require a controller of order $2n-2$, a controller of order $2n+2$ is chosen. The four extra degrees of freedom are used to force some of the poles of the compensator onto the the $j\omega$-axis. Adaptation is employed to ensure that the poles converge to the desired locations on the $j\omega$-axis, thereby rejecting the disturbance.

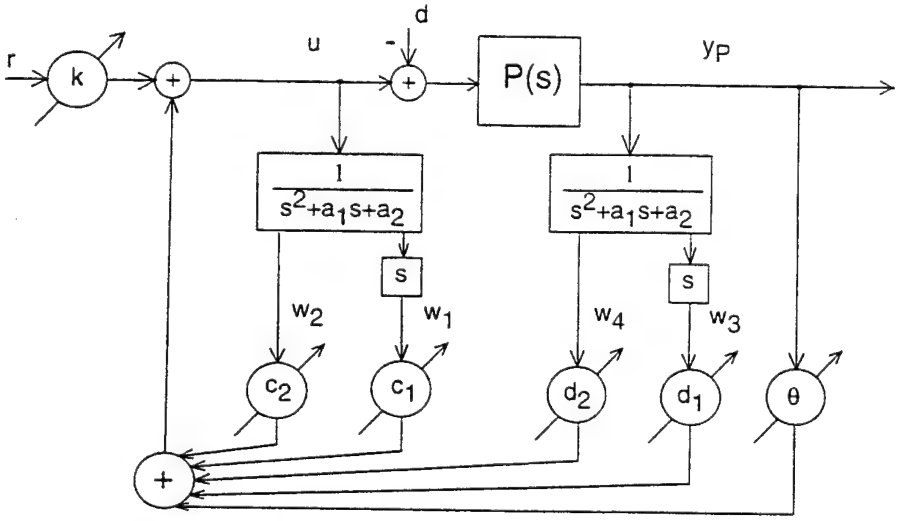

Figure 1: Adaptive Scheme Based on the Internal Model Principle

For illustration, consider the scheme of [2], as shown in Fig. 1. This figure is identical to that on p. 302 of [2], except for a difference in the sign convention for the disturbance. This particular realization assumes that $P(s)$ is a first-order plant of the form

$$P(s) = \frac{y_P(s)}{u(s)} = \frac{k_P}{s + a_P}. \tag{3}$$

where the parameters k_P and a_P are unknown. The scheme can be extended to higher-order plants by increasing the order of the filters and the number of adaptive parameters. The control objective is to track the output of a reference model $M(s)$ with reference input $r(t)$ and

$$M(s) = \frac{y_M(s)}{r(s)} = \frac{k_M}{s + a_M}. \tag{4}$$

In this system, k_M and a_M are design parameters. The control law is given by

$$u(t) = k(t)r(t) + c_1(t)w_1(t) + c_2(t)w_2(t) + \theta(t)y_p(t) + d_1(t)w_3(t) + d_2(t)w_4(t). \tag{5}$$

The signals w_1, w_2, w_3, w_4 are determined by

$$w_1(s) = \frac{s}{s^2 + a_1 s + a_2}\, u(s), \qquad w_2(s) = \frac{1}{s^2 + a_1 s + a_2}\, u(s), \tag{6}$$

$$w_3(s) = \frac{s}{s^2 + a_1 s + a_2}\, y_P(s), \quad \text{and} \quad w_4(s) = \frac{1}{s^2 + a_1 s + a_2}\, y_P(s). \tag{7}$$

The adaptive parameters are updated according to

$$\begin{aligned} \dot{k}(t) &= -ge_0(t)r(t), \quad \dot{\theta}(t) = -ge_0(t)y_P(t), \quad \dot{c}_2(t) = -ge_0(t)w_2(t), \\ \dot{d}_1(t) &= -ge_0(t)w_3(t), \quad \dot{d}_2(t) = -ge_0(t)w_4(t). \end{aligned} \tag{8}$$

where $g > 0$ is an adjustable adaptation gain and

$$e_0(t) = y_P(t) - y_M(t) \tag{9}$$

is the output error.

For a first-order plant without an input disturbance, a similar compensator without the signals w_1, w_2, w_3, and w_4 can be used, and only the two parameters k and θ are adapted. In the presence of the disturbance, the compensator is the same as the one that would have been used for a third-order plant without the disturbance. In this system, however, the parameter c_1 is fixed at a value such that the loop transfer function for constant parameters, given by

$$L(s) = \frac{s^2 + a_1 s + a_2}{s^2 + (a_1 - c_1)s + (a_2 - c_2)} \cdot \frac{k_P}{s + a_P} \cdot \frac{\theta(s^2 + a_1 s + a_2) + (d_1 s + d_2)}{s^2 + a_1 s + a_2}, \tag{10}$$

has a pair of poles on the $j\omega$-axis. This will be the case when

$$c_1^* = a_1, \tag{11}$$

$$c_2^* = a_2 - \omega_1^2. \tag{12}$$

In the algorithm, c_1 is chosen according to (11), while c_2 is updated according to (8) as the frequency ω_1 is unknown.

The nominal values for the other parameters are obtained by calculating the closed-loop transfer function for constant parameters, which is

$$\frac{y_P(s)}{r(s)} = k \cdot \frac{s^2 + a_1 s + a_2}{s^2 + (a_1 - c_1)s + (a_2 - c_2)} \cdot \frac{k_P}{s + a_P} \cdot (1 - L(s))^{-1}. \tag{13}$$

The transfer function matches the reference model transfer function (4) for the nominal parameter values

$$k^* = \frac{k_M}{k_P}, \quad \theta^* = \frac{a_P - a_M - a_1}{k_P}, \quad d_1^* = \frac{a_1^2 - c_2 - a_1 a_P}{k_P}, \quad d_2^* = \frac{a_2 a_1 - c_2 a_P}{k_P}. \tag{14}$$

2.2 Simulation Results

The scheme has several useful features, including its capabilities to control unknown plants, to stabilize unknown plants, to reject disturbances of unknown frequencies, and to track arbitrary reference inputs. These properties are explored in a simulation in which $k_P = 2$, $a_P = -1$, $k_M = 1$, $a_M = 2$, $a_1 = 3$, $a_2 = 2$, and $g = 1$. The reference input is $r(t) = 10$ and the disturbance is $d(t) = -2\sin(5t)$. All the initial conditions are zero, except for $y_M(0) = 1$ and $k(0) = 0.5$. In addition, the estimate k is kept constant for the entire simulation instead of being updated and is

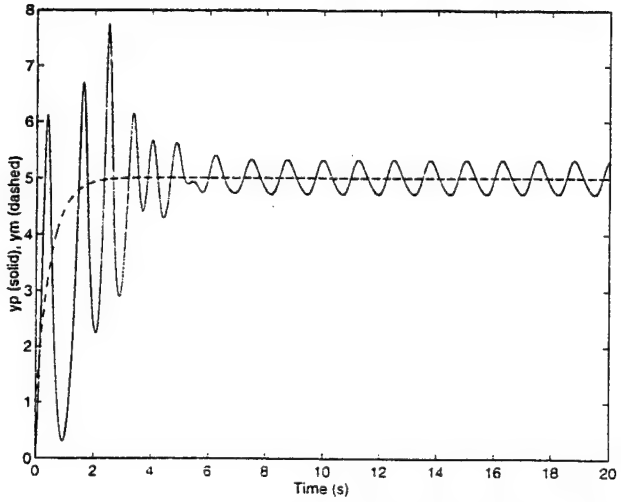

Figure 2: Plant Output and Reference Model Output

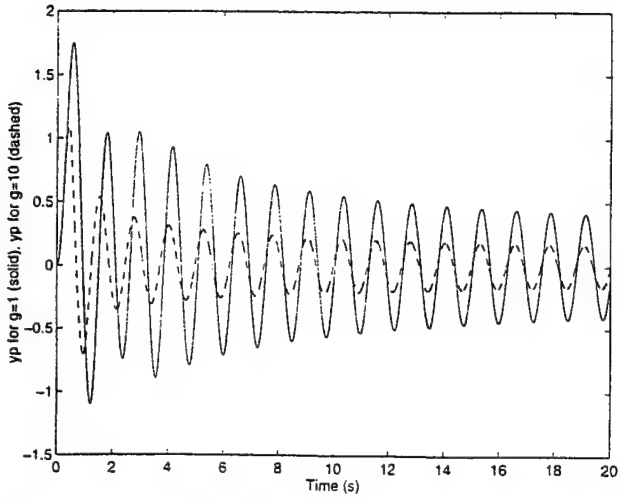

Figure 3: Plant Output for Zero Reference Input and Two Values of the Gain

equal to the nominal value $k^* = 0.5$. These conditions are the same as those used in a simulation from [2]. Indeed, Fig. 2 is essentially a replica of a figure found on p. 302 of [2]. In Fig. 2, the solid line is the plant output y_P and the dashed line is the reference model output y_M. Note that the plant is unstable, so that both stabilization of the unknown plant and tracking of the reference input are achieved. However, rejection of the disturbance is not obtained; a large oscillation remains after 6 seconds.

Fig. 3 shows a similar simulation, except that $r(t) = 0$ and $y_M(0) = 0$. The output is shown for two values of the gain: the solid line is for $g = 1$ and the dashed line is for $g = 10$. The output again exhibits a large residual error, although it appears to converge to zero. A close examination at the responses indicates that the convergence of the amplitude of the error is approximately of the order of $1/\sqrt{t}$, which is very slow. Moreover, increasing the adaptation gain g by a factor of 10 only reduces the error by a factor of two, as seen in Fig. 3. Fig. 4 shows the adaptive parameters c_2 (left) and θ (right). Again, the solid lines are for $g = 1$ and the dashed lines are for $g = 10$. One finds that c_2 remains far from the nominal value $c_2^* = -23$ which would guarantee the rejection of the disturbance by virtue of the internal model principle. On the other hand, the feedback gain θ far exceeds its nominal value of $\theta^* = -3$. It appears that the error is reduced through a large feedback gain θ rather than through the convergence of c_2 to its nominal value in this situation. Thus, the principle upon which the scheme was developed is not realized.

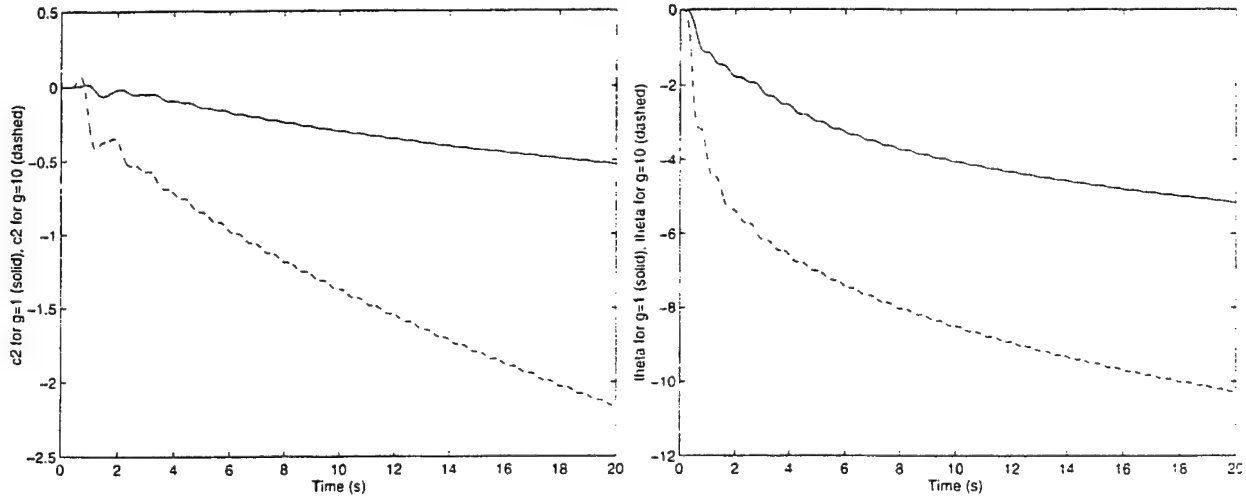

Figure 4: Responses of c_2 (left) and θ (right)

The lack of good convergence properties for this scheme may be expected to lead to poor robustness properties in the presence of unmodelled plant dynamics. Indeed, Fig. 5 shows the result of a simulation in which the plant is

$$P(s) = \frac{k_P}{s + a_P} \cdot \frac{229}{s^2 + 30s + 229}. \tag{15}$$

The unmodelled dynamics represented by the second term in (15) are those of the well-known "Rohrs examples" ([7], [8]) and represent relatively mild modifications to the original plant. Yet, the output of the plant, shown on the left of Fig. 5, rapidly diverges. On the right of the figure is the trajectory of the adaptive parameter θ, which is seen to reach large negative values and is likely to be the cause of the instability.

2.3 Alternative Design Options

It is possible to alleviate some of the problems discussed in section 2.2 by making alternative design choices with the algorithm's structure. For the convergence problems, one possible remedy is the use of richer reference inputs. Unfortunately, simulations in which $r(t)$ consisted of several sinusoids did not produce good results. Moreover, zero reference inputs are common in active noise and vibration control problems. A better option would be to replace

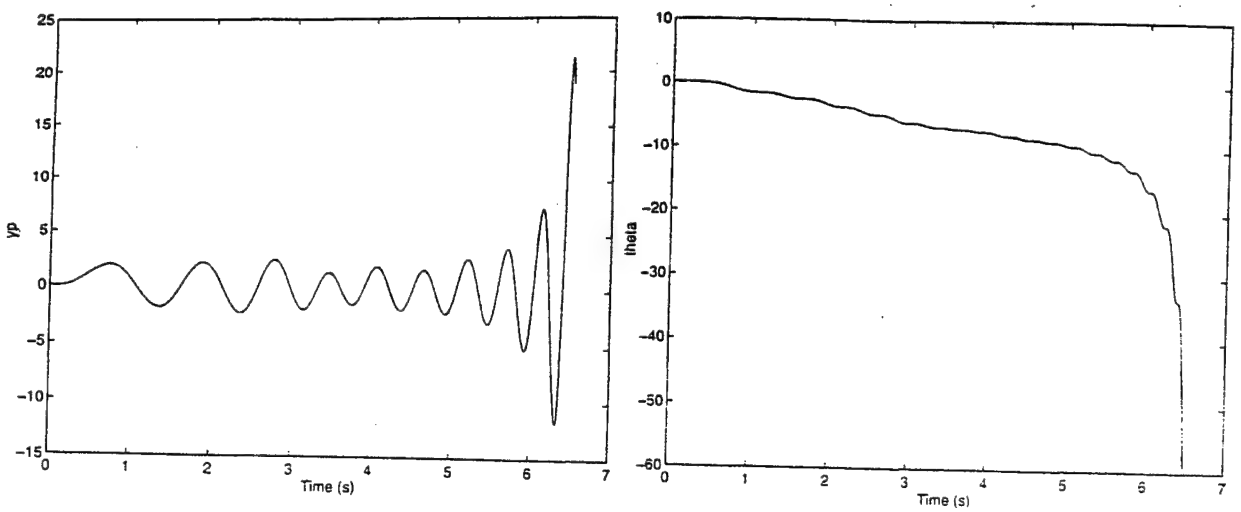

Figure 5: Responses of y_P (left) and θ (right) with Unmodelled Dynamics

the pseudo-gradient algorithm (8) by a least-squares algorithm. However, this change would require a different adaptation structure with a more complicated adaptation rule.

As for the robustness problems, it is noted that the control system is not robust for settings at the nominal parameter values. In the example of Section 2.2, the loop transfer function with the nominal parameters provides only a gain margin of 1.51 dB and a phase margin of 18 degrees. These low margins can be attributed to the fact that the plant is unstable and that poles at $s = 1$, $s = \pm 5j$ are moved to distant locations at $s = -1$, $s = -1$, $s = -2$. A higher level of robustness is obtained if the open-loop plant is stable and if the observer poles (the roots of $s^2 + a_1 s + a_2$) are placed at values closer to $s = \pm j\omega_1$. This placement requires some prior knowledge about the value of the frequency of the disturbance.

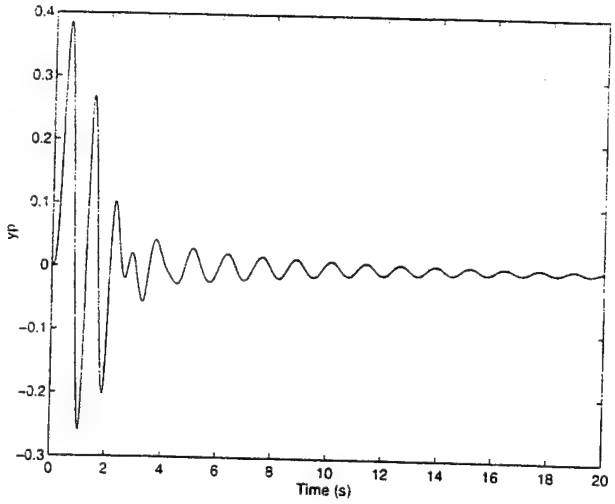

Figure 6: Plant Output with Unmodelled Dynamics and Improved Design

Simulations were performed with the plant pole and the reference model pole at $s = -1$ and the observer poles at $s = -1 \pm 5j$. In that case, the nominal gain margin is infinite and the phase margin is equal to 90 degrees. The adaptive parameters were also initialized at values closer to the nominal values. Specifically, the initial parameters corresponded to the nominal plant parameters and to a disturbance frequency equal to 90% of the true frequency, or 4.5 rad/s. The output y_P is shown in Fig. 6, indicating that the algorithm is stable in this situation. Small values

of the error are also reached. However, a large adaptation gain of $g = 100$ had to be used to obtain this result. The large gain may pose potential problems in the presence of measurement noise. Nevertheless, the plot shows that much better robustness properties can be achieved with proper design and initialization, provided that prior information is available about the plant and about the disturbance.

3. MAGNITUDE/PHASE LOCKED LOOP SCHEME

3.1 Algorithm

In this section, we discuss another algorithm that is based on a magnitude/phase locked loop concept. The scheme is shown in Fig. 7. It is similar to a phase locked loop, except that the disturbance signal is matched both in phase and magnitude. The scheme is quite different from the scheme based on the internal model principle in Fig. 1. The reference input is absent, *i.e.*, it is zero. In addition, the plant is assumed to be stable.

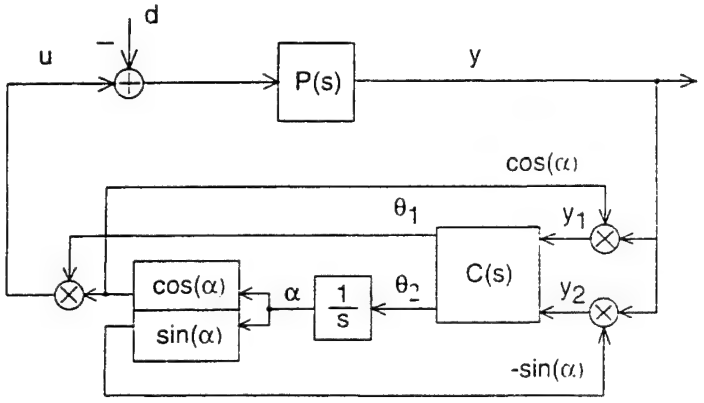

Figure 7: Magnitude/Phase Locked Loop Scheme

In the figure, θ_1 is the estimate of the magnitude of the disturbance signal, and θ_2 is the estimate of its instantaneous frequency. Moreover, α is the estimate of the phase of the disturbance signal and is the integral of the estimate of the frequency θ_2. The equations for the control algorithm are

$$\begin{aligned} u(t) &= \theta_1(t)\cos(\alpha(t)), \\ \dot{\alpha}(t) &= \theta_2(t), \\ y_1(t) &= y(t)\cos(\alpha(t)), \\ y_2(t) &= -y(t)\sin(\alpha(t)). \end{aligned} \tag{16}$$

The last part of the algorithm is the transfer function matrix relating the signals y_1 and y_2 to the parameters θ_1 and θ_2. Its design is based on the following result.

Fact: Assume that

- The signals θ_1 and θ_2 vary sufficiently slowly that the output of the plant to the signal u can be approximated by the steady-state response for a sinusoidal input with frequency θ_2.
- The instantaneous frequency θ_2 is close to ω_1, such that $P(j\theta_2)$ can be replaced by $P(j\omega_1)$.

Then, considering low-frequency components only, the two signals $y_1(t)$ and $y_2(t)$ are approximately given by

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = G \begin{bmatrix} \theta_1(t) - d_1\cos(\alpha(t) - \alpha_d(t)) \\ d_1\sin(\alpha(t) - \alpha_d(t)) \end{bmatrix}. \tag{17}$$

where G is defined by

$$G = \frac{1}{2}\begin{bmatrix} P_R & -P_I \\ P_I & P_R \end{bmatrix}, \quad P_R = Re[P(j\omega_1)], \quad P_I = Im[P(j\omega_1)]. \tag{18}$$

Proof: Under the assumptions, the output of the plant is given by

$$y(t) = P_R\theta_1(t)\cos(\alpha(t)) - P_I\theta_1(t)\sin(\alpha(t)) - P_R d_1\cos(\alpha_d(t)) + P_I d_1\sin(\alpha_d(t)). \tag{19}$$

Keeping only the low-frequency components of the signals y_1 and y_2, we find that

$$\begin{aligned} y_1(t) &= \frac{1}{2}P_R\theta_1(t) - \frac{1}{2}P_R d_1\cos(\alpha(t)-\alpha_d(t)) - \frac{1}{2}P_I d_1\sin(\alpha(t)-\alpha_d(t)), \\ y_2(t) &= \frac{1}{2}P_I\theta_1(t) + \frac{1}{2}P_R d_1\sin(\alpha(t)-\alpha_d(t)) - \frac{1}{2}P_I d_1\cos(\alpha(t)-\alpha_d(t)), \end{aligned} \tag{20}$$

and the result is obtained.

The elimination of the high-frequency components within the system can be achieved by low-pass filtering of the signals $y_1(t)$ and $y_2(t)$. In the algorithm discussed in this paper, the signals are applied to a compensator which is low-pass in nature. Although the filtering is far from ideal, simulations show that the approximation is satisfactory for the compensator design without the need for additional filtering.

Equation (17) can be viewed as a new description of the plant, with two inputs θ_1 and θ_2 and two outputs y_1 and y_2. Recall that α is the integral of the signal θ_2. A linear system is obtained if the phase error $\alpha - \alpha_d$ is small. The system is then described by

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = G \begin{bmatrix} \theta_1(t) - d_1 \\ d_1(\alpha(t) - \alpha_d(t)) \end{bmatrix}. \tag{21}$$

Defining two variables x_1 and x_2 as

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = G^{-1}\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \tag{22}$$

one finds that

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \theta_1(t) - d_1 \\ d_1\left(\int_0^t (\theta_2(\sigma) - \omega_1)d\sigma + \alpha(0) - \alpha_d(0)\right) \end{bmatrix}. \tag{23}$$

As a result, the dynamics of the system from the parameters θ_1 and θ_2 to the variables x_1 and x_2 are decoupled from one another and are those of a gain of 1 and of an integrator with a gain d_1, respectively. The unknown parameters d_1, ω_1, $\alpha_d(0)$ act as constant disturbances at the input of the system.

Simulations have shown that satisfactory performance is obtained when the compensator is designed using linear techniques. We choose it to be the cascade of the transformation (22) and control laws of the form

$$\begin{aligned} \theta_1 &= \frac{C_1(s)}{s}[x_1], \\ \theta_2 &= \frac{C_2(s)}{s}[x_2], \end{aligned} \tag{24}$$

where the integrators are included to reject the disturbances composed of d_1 and $\alpha_d(0)$. The transfer functions $C_1(s)$ and $C_2(s)$ must guarantee the closed-loop stability of the two systems $P_1(s) = 1$ and $P_2(s) = d_1/s$. A simple choice is

$$C_1(s) = -g_1, \qquad C_2(s) = -g_2\frac{s+a}{s+b}. \tag{25}$$

The closed-loop pole for the first system is placed at some desired value, say $-s_d$, and the closed-loop poles for the second system are placed at $-s_d$, $-s_d \pm js_d$, by setting

$$g_1 = s_d, \quad a = \frac{s_d}{2}, \quad b = 3s_d, \quad g_2 = \frac{4{s_d}^2}{d_1}. \tag{26}$$

Because the magnitude of the disturbance d_1 acts as a gain in the second transfer function, the gain of the compensator $C_2(s)$ depends on an estimate of the magnitude. The algorithm also requires an estimate of the matrix G, *i.e.*, an estimate of the frequency response of the plant around ω_1. Often, the uncertainty in the frequency of the disturbance is small enough that errors in the G matrix are inconsequential. Alternatively, one could use for G the value of the frequency response at the estimated frequency $\dot{\alpha} = \theta_2$. In either case, knowledge of the frequency response of the plant in the frequency range of interest is required.

3.2 Simulation Results

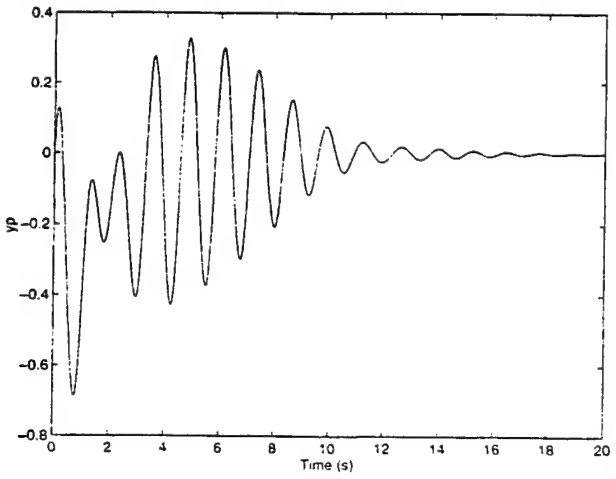

Figure 8: Plant Output

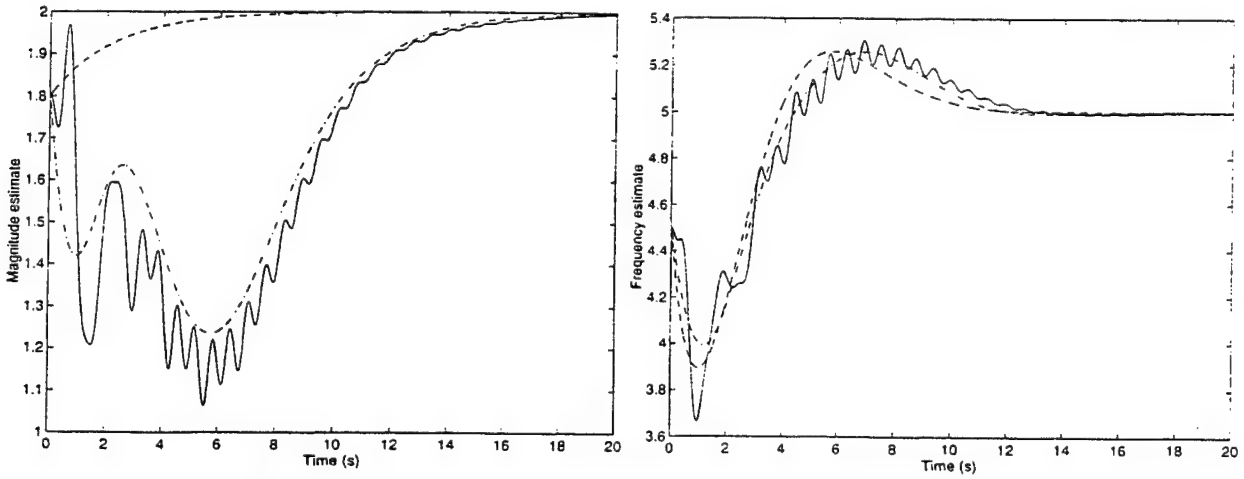

Figure 9: Magnitude Estimate (left) and Frequency Estimate (right)

The performance of the algorithm was evaluated via simulation. We consider the same case as in Fig. 6, with $P(s) = 1/(s+1)$, $\omega_1 = 5$, $d_1 = 2$, and $\alpha_d(0) = -90$ degrees. The initial states are all zero except for $\theta_1(0) = 1.8$ and $\theta_2(0) = 4.5$ rad/s, corresponding to 10% initial errors in magnitude and frequency. The matrix G is calculated using the initial estimate of the frequency. Since $\alpha(0) = 0$ degrees and $\alpha_d(0) = -90$ degrees, the system is initialized with a phase error out of the linear region. Fig. 8 shows the output of the system, which is found to rapidly converge to

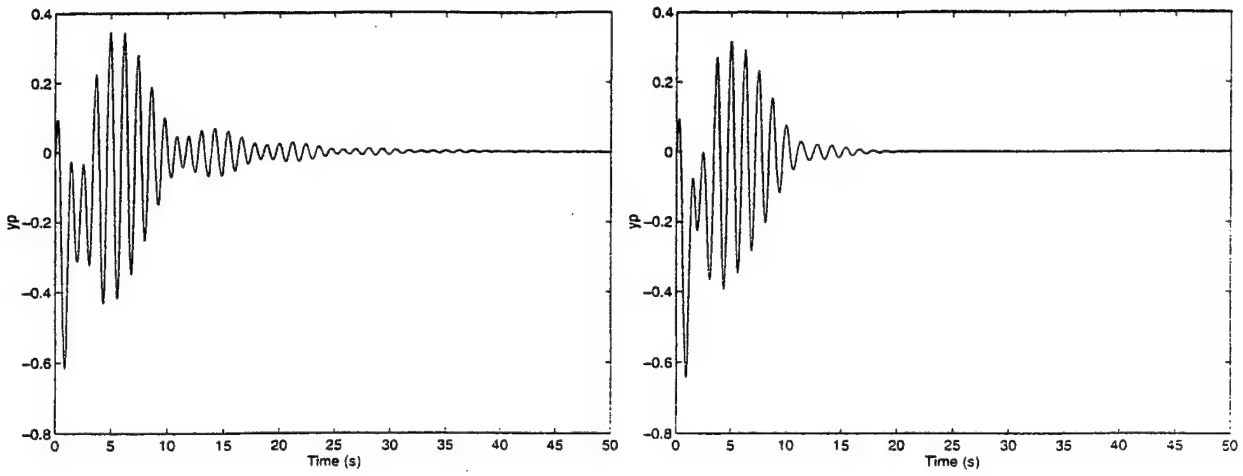

Figure 10: Plant Output with Unmodelled Dynamics

zero. Fig. 9 shows the magnitude estimate θ_1 on the left and the frequency estimate θ_2 on the right. The solid lines are the estimates, the dashed lines are obtained by integrating the equations with the linear approximation in (21) and the dot-dashed lines are obtained using the nonlinear approximation in (17). The nonlinear approximation is found to represent very well the transient response of the system, although the linear approximation is sufficient to produce a successful design.

Fig. 10 on the left shows the controller output in the presence of the unmodelled dynamics in (15). Stability is preserved, although the convergence of the error to zero is slower. On the right of Fig. 10 is a similar response, except that the G matrix is obtained by using the true frequency response of the plant at the initial estimate of the frequency (4.5 rad/s). One finds that the speed of convergence is restored if the true frequency response is used. The frequency response can, in general, be obtained in a tuning phase of the algorithm.

3.3 Experimental Results

The scheme was implemented in an experimental active noise control system under development at the University of Utah. This system employs a Motorola DSP96002 32-bit floating-point digital signal processor. The sampling rate for all signals was 8 kHz. A single bookshelf speaker with a 4-inch low-frequency driver located approximately 2 ft away from the error microphone generated a pure sinusoidal tone at a frequency of approximately 150 Hz. The microphone signal was passed through an anti-aliasing filter and sampled by a self-calibrating 16-bit analog-to-digital converter before being sent to the DSP system. The noise cancellation signal generated by this system was sent to a second speaker placed approximately 1 ft away from the microphone. Only the error sensing microphone signal was provided to the algorithm for the controller calculations. The frequency response of the plant was calculated from a 50-tap finite impulse response filter model. The coefficients of this filter were determined in a brief identification phase in which a white noise signal was applied to the noise cancellation speaker. As expected, a delay of approximately 8 samples was observed in this impulse response.

A straightforward transposition of the phase-locked loop algorithm to the discrete-time case was implemented in the DSP-based system (*cf.* [6]). Fig. 11 shows the magnitude of the error signal, as measured by the microphone in the experiment and plotted in dB. For the first 0.5 seconds of the experiment, the algorithm was not engaged. Once adaptation commenced, convergence occurred within 1 second and yielded approximately 25 dB reduction in the sound output at the error microphone.

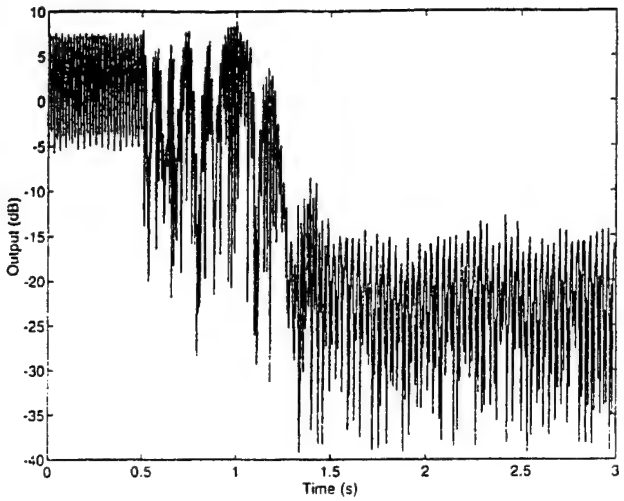

Figure 11: Experimental Plant Output

4. CONCLUSIONS

The scheme based on the adaptive internal model principle has powerful capabilities by enabling the stabilization of unstable plants and the tracking of reference inputs. The scheme also tolerates, in theory, a complete uncertainty about the plant parameters and about the disturbance frequency. In practice, however, significant prior information may be required to achieve adequate robustness margins. It is also difficult to obtain reasonable convergence rates with a pseudo-gradient algorithm. Implementation of the algorithm was not attempted because of these concerns and because it is difficult to apply the algorithm to the case where the order of the plant is high, or where there is a variation in time delay. The discrete-time implementation for a finite impulse response plant model with 50 coefficients would require 104 adaptive parameters.

The magnitude/phase locked loop scheme was found to be easier to implement and to have good convergence and robustness properties for an active noise control task. A linear approximation permitted a convenient selection of the design parameters for this system. The scheme assumes that the plant is stable and that the reference input is zero, two conditions that are reasonable in noise control applications. To use this scheme effectively, some prior information about the frequency response of the plant as well as the magnitude and frequency of the disturbance is required.

ACKNOWLEDGEMENTS

This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred. The authors would like to thank Shayne Messerly for his help in performing the experiments reported in the paper. The Motorola Corporation, and Bob Bergeler in particular, are gratefully acknowledged for donating the DSP96000 Development System used in this research.

REFERENCES

[1] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, Wiley-Interscience, New York, NY, 1996.

[2] K.S. Narendra and A. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] G. Feng and M. Palaniswamy, "Unified Treatment of Internal Model Principle Based Adaptive Control Algorithm," *Int. J. Control*, vol. 54, no. 4, pp. 883-901, 1991.

[4] G. Feng and M. Palaniswamy, "A Stable Adaptive Implementation of the Internal Model Principle," *IEEE Trans. on Automatic Control*, vol. 37, no. 8, pp. 1220-1225, 1992.

[5] M. Bodson and S.C. Douglas, "Rejection of Disturbances with a Large Sinusoidal Component of Unknown Frequency," *Proc. of the SPIE Symposium on Smart Structures and Materials*, San Diego, CA, vol. 2715, pp. 64-75, 1996.

[6] M. Bodson, "Cancellation of Sinusoidal Disturbances with Unknown Frequency," *Proc. of the Ninth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, pp. 174-179, 1996.

[7] C.E. Rohrs, L. Valavani, M. Athans, and G. Stein, "Robustness of Adaptive Control Algorithms in the Presence of Unmodeled Dynamics," *Proc. of the 21st IEEE Conference on Decision and Control*, Florida, pp. 3-11, 1982.

[8] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Algorithms for Feedback Active Noise Control and Narrowband Disturbances

M. Bodson
Department of Electrical Engineering, University of Utah,
Salt Lake City, UT 84112, U.S.A.
e-mail : bodson@ee.utah.edu

Abstract

The paper considers the problem of active noise control without feedforward sensors and for narrowband disturbances. Attention is focused on sinusoidal disturbances, although the methods under consideration may be extended to general periodic disturbances. The frequency of the disturbance is assumed to be unknown. Three approaches are reviewed in the paper: the adaptive implementation of the internal model principle, the combination of adaptive cancellation techniques with frequency estimation algorithms, and the direct adaptation of the magnitude, frequency, and phase of the control input. The respective advantages and disadvantages of the approaches are discussed in detail.

1. Introduction

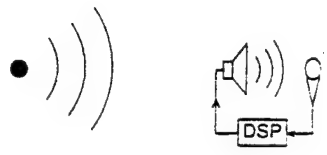

Figure 1: Feedback Active Noise Control

A single-channel active noise control system is shown in Fig. 1. A speaker produces a sound wave that is adjusted to cancel the noise generated by an external source. The sound intensity is sensed by a microphone at some location to be made quiet, and the signal is applied to a digital signal processing system (DSP) to compute the required speaker signal. No sensor is placed on the machine that is the source of the disturbance, or on its path, so that feedforward compensation is not possible.

The disturbance is assumed to be periodic, which is a common situation when the source of the noise is a rotating machine. The overall setup is then referred to as feedback active noise control for narrowband disturbances [1]. The frequency of the disturbance is not assumed to be known, which poses significant difficulties and limits the number of techniques that may be applied.

Mathematically, the problem is stated as follows. The transformation from the speaker to the microphone is assumed to be that of a stable linear time-invariant system with transfer function $P(s)$. The effect of the noise source is assumed to be additive, and the disturbance is reflected to the location of the speaker. The system is then modelled by

$$y(s) = P(s)(u(s) - d(s)), \qquad (1)$$

where $y(s)$, $u(s)$, and $d(s)$ are the Laplace transforms of the microphone signal, of the speaker signal, and of the equivalent noise signal at the speaker location, respectively. The problem may be viewed as a standard feedback control problem, with a zero reference input and a disturbance acting at the input of the plant. $P(s)$ is the plant transfer function, $y(t)$ is the plant output, $u(t)$ is the control input, and $d(t)$ is the disturbance.

The goal of the control system is to generate $u(t)$ such that $y(t) \to 0$ as $t \to \infty$. The objective would be achieved if $u(t) = d(t)$, but the disturbance $d(t)$ is not known or measured, except through its effect at the output of the system. The disturbance is of the form

$$\begin{aligned} d(t) &= \sum_{k=1}^{n} d_k \cos(\alpha_{d,k}(t)), \\ \dot{\alpha}_{d,k}(t) &= k\omega_1, \quad k = 1, ..., n. \end{aligned} \qquad (2)$$

The frequency of the disturbance, ω_1, is unknown. The parameters d_k and $\alpha_{d,k}(0)$ specify the magnitudes and the phases of the harmonic components, and are also unknown. The number of harmonics n is assumed to be finite and known. Certain values of d_k may be assumed to be zero. For simplicity of presentation, we will consider primarily the case where only the fundamental is present (sinusoidal disturbances).

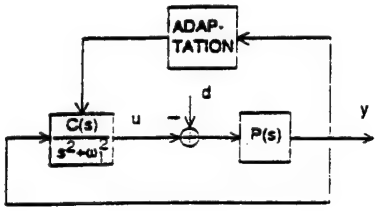

Figure 2: Adaptive Internal Model Principle

Three approaches are considered in the paper. The first approach is based on the adaptive implementation of the internal model principle. The concept is represented in Fig. 2. The internal model principle states that sinusoidal disturbances of frequency ω_1 will be perfectly rejected if the feedback compensator has poles located at $s = \pm j\omega_1$ in the s-plane. The name comes from the fact that, for most initial conditions and in open-loop, such compensator generates sinusoidal signals of frequency ω_1. The adaptive implementation of the principle, illustrated in Fig. 2, might seem straightforward. However, the poles and zeros of $C(s)$ need to be adjusted as ω_1 varies to maintain stability.

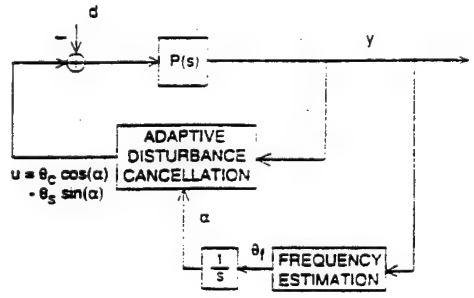

Figure 3: Indirect Approach

The second approach under consideration is shown in Fig. 3 and combines an adaptive algorithm for the cancellation of periodic disturbances of known frequency together with a frequency estimation algorithm. The adaptive cancellation algorithm estimates θ_c and θ_s, the *cos* and *sin* components of the control input. The frequency estimation algorithm provides an estimate θ_f of the frequency of the disturbance, which is integrated to obtain a phase signal α. Under ideal conditions, the phase is equal to the phase of the disturbance up to a constant angle that is estimated through the parameters θ_c and θ_s. The approach is called indirect, because an intermediate frequency estimation step is performed, independently of the adaptation to the magnitude and phase of the disturbance.

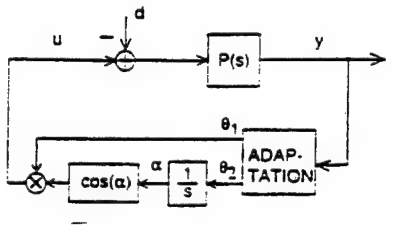

Figure 4: Direct Approach

Fig. 4 shows the concept of a scheme where the estimation of the magnitude, frequency, and phase of the disturbance are all performed together, with a single adaptation mechanism. θ_1 is the estimate of the magnitude of the disturbance, θ_2 is the estimate of its frequency, and α is the estimate of its phase. The objective of the adaptation scheme is to provide a stable closed-loop system and convergence of $y(t)$ to zero.

In the remaining of the paper, specific implementations of the three approaches are presented, and their respective advantages and disadvantages are discussed. Details of the algorithms are left to the papers cited as references.

2. Adaptive Internal Model Principle

The internal model principle has been considered as a constraint on adaptive control schemes for unknown plants. Fig. 5 is an example found in Narendra & Annaswamy's book [2], and shows

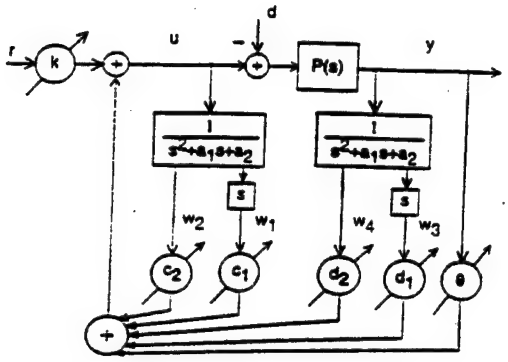

Figure 5: Narendra & Annaswamy's Example

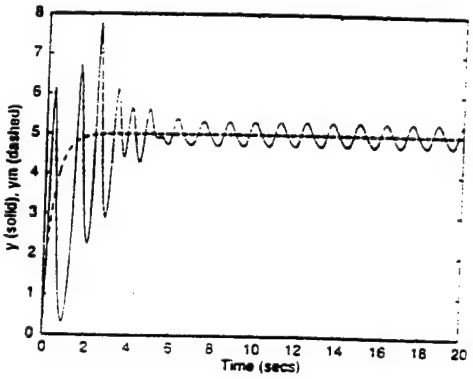

Figure 6: Output with N.&A.'s Example

the control structure of an adaptive control algorithm for a first-order system with unknown parameters and a sinusoidal disturbance of unknown frequency. The parameters k, c_2, d_1, d_2, and θ are updated according to a standard gradient algorithm, while the constraint $c_1 = a_1$ is applied so that the feedback loop has a pair of poles on the $j\omega$-axis. Alternative algorithms based on the internal model principle may be found in [3], [4]. In these algorithms, the problem of adapting to the frequency of the disturbance is solved together with the problem of adapting to uncertain plant parameters. In addition, tracking of reference inputs is possible, and unstable plants may be stabilized. In other words, the algorithms solve a much more general problem than the one stated earlier.

The schemes, however, are not without serious drawbacks. Convergence and robustness properties are generally poor. Fig. 6 shows the plant output resulting from a simulation of the example of Fig. 5 (this figure is identical to a figure found in [2]). The dashed line shows the output of a reference model, which the plant output is supposed to follow. A large residual error is visible. While the theoretical results guarantee the stability of the adaptive system and the convergence of the error to zero, the convergence is extremely slow. It also turns out that instabilities result from very small amounts of unmodelled dynamics, and that the schemes based on the internal model principle require a large number of parameters for systems with high order and large time delays. For a detailed discussion of these issues, see [5].

Overall, the approach based on the internal model principle offers some powerful capabilities, but these capabilities are not exploited in active noise control problems. Further, weaknesses severely limit the applicability of the methods. Stability and robustness issues need to be resolved before these schemes can be practically useful.

3. Indirect Approach

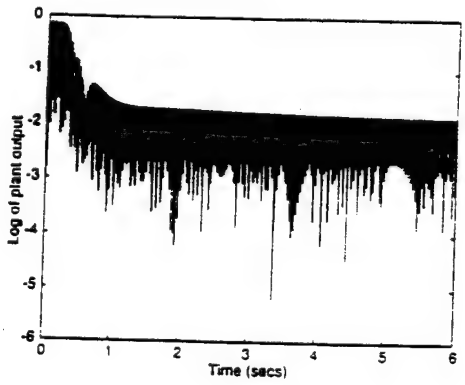

Figure 7: Output with Indirect Scheme

The indirect approach to periodic disturbance cancellation is probably most intuitive. Algorithms for the adaptive cancellation of periodic disturbances of known frequency are widely available and work well (see, for example, [6], [7]). Frequency estimation algorithms are also numerous. In [8], the adaptive cancellation scheme of [6] was combined with the frequency estimation algorithm of [9]. An unexpected problem was

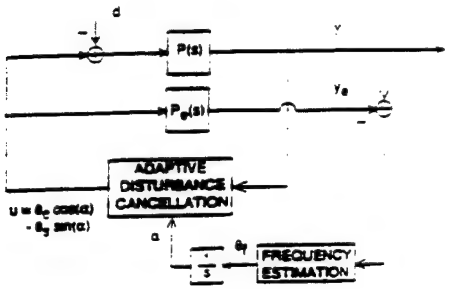

Figure 8: Modified Indirect Approach

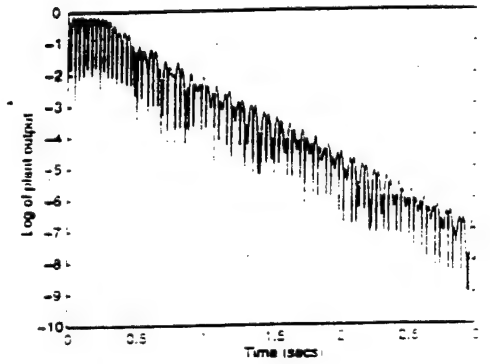

Figure 9: Output with Modified Indirect Scheme

discovered, in that the convergence of the combined scheme was found to be slow. This problem is illustrated in Fig. 7, which shows the log of the plant output for a simulation of the indirect scheme. As one can see, there is rapid convergence of the output signal towards zero, but the convergence slows down considerably after a second. The problem was traced to the fact that, as the adaptive algorithm cancels the effect of the disturbance, it also eliminates the signal upon which frequency estimation is based, resulting in slow convergence.

While the problem may seem to be tied to the indirect approach, that is not the case. A modified indirect approach, shown in Fig. 8, relies on a signal for frequency estimation that does not converge to zero. The signal is obtained by estimating the plant output y_e that is due to the control input using an estimate of the plant transfer function $P_e(s)$, and subtracting the signal from the plant output. If $P_e(s) = P(s)$, the resulting signal is equal to the output that would be observed if no control signal was applied to the plant, i.e., the component that is due to the disturbance alone. Fig. 9 shows the plant output that results from the modification applied to the scheme considered earlier. Exponential convergence of the error signal is observed. One might be concerned that the success of such a modified scheme would be highly dependent upon the accuracy of the estimate of the plant transfer function. However, simulations have not confirmed such a concern.

Overall, the modified indirect approach is very flexible. The modularity of the design allows one to select the two components independently, and each from a large number of possible algorithms. On the other hand, analysis is difficult, and interactions between the two components may be complex and incompletely understood.

4. Direct Approach

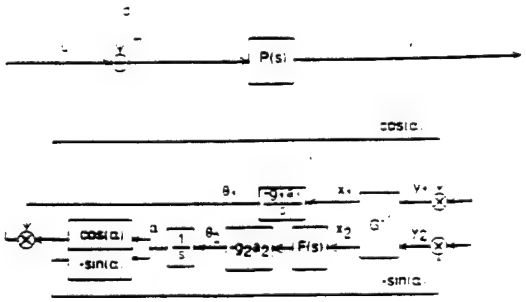

Figure 10: Direct Scheme

Fig. 10 shows the block diagram of the direct algorithm proposed in [10]. It is a fairly simple scheme, which can be translated in a few lines of code. In the figure, G is a 2×2 matrix given by

$$G = \begin{pmatrix} P_R & -P_I \\ P_I & P_R \end{pmatrix}, \qquad (3)$$

where P_R and P_I are the real part and imaginary part of the plant's frequency response at ω_1. The filter $F(s)$ is a lead/lag filter with integral action, that is $F(s) = (s - a)/(s^2 - bs)$. The variables $g_1 a_1$, $g_2 a_2$ are adjustable gains. The free parameters are chosen to ensure satisfactory convergence properties.

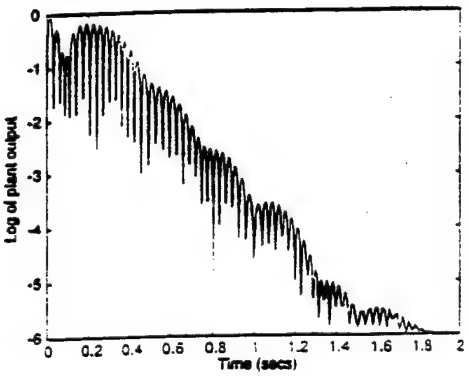

Figure 11: Plant Output

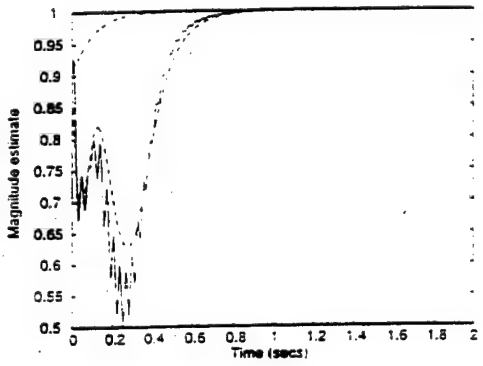

Figure 12: Magnitude Estimate

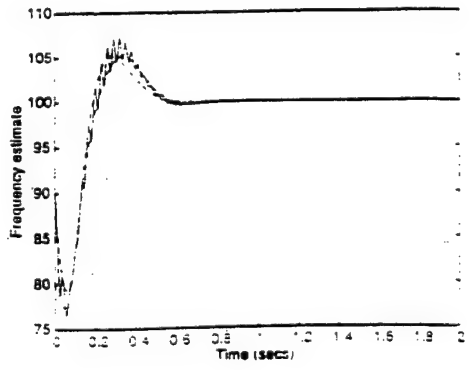

Figure 13: Frequency Estimate

The results of simulations for this scheme are shown in Fig. 11, Fig. 12, and Fig. 13. The parameters of the simulation are given in [8]. The log of the plant output, shown on Fig. 11, exhibits a similar exponential convergence as that of the modified indirect scheme. Fig. 12 shows the estimate of the magnitude θ_1, and Fig. 13 shows the estimate of the frequency θ_2. Both parameters are found to converge rapidly. The dashed line and the dot-dashed line show the results of simulations of an approximate nonlinear time-invariant system and an approximate linear time-invariant system, respectively. The approximations are found to be accurate, especially for the nonlinear system, and were obtained using an analysis similar to the averaging analysis that is sometimes applied to adaptive systems [11].

The nonlinear approximation is found to capture significant nonlinear dynamics of the magnitude parameter. The dynamics of the direct scheme have been found to be similar to those of phase-locked loops. In fact, the scheme may be viewed as a "magnitude/phase-locked loop". The linear approximation, although less accurate, is useful for the selection of the algorithm's parameters. Specifically, the parameters may be selected to achieve pre-determined convergence rates. Further analysis using the linear approximation leads to predictions of the effect of slow variations of the magnitude and frequency of the disturbance [10], and of the effect of noise on the measurements [12].

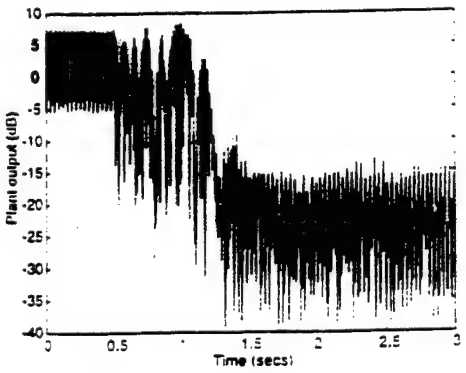

Figure 14: Microphone Signal (Experiment)

The scheme was implemented on an experimental active noise control system developed at the University of Utah. The algorithm was coded

in assembly language on a Motorola DSP96002 32-bit floating-point digital signal processor. The sampling rate was set at 8 kHz. A single bookshelf speaker with a 4-inch low-frequency driver, located approximately 2 ft away from the error microphone, generated a periodic signal constituting the noise source. The microphone signal was passed through an anti-aliasing filter and sampled by a self-calibrating 16-bit analog-to-digital converter before being sent to the DSP system. The noise cancellation signal was sent to a second speaker placed approximately 1 ft away from the microphone. Only the error sensing microphone signal was provided to the algorithm for the controller calculations.

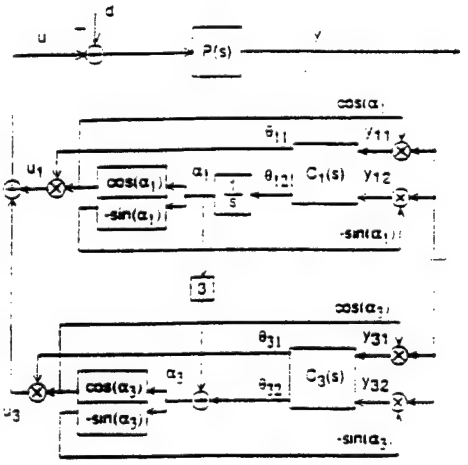

Figure 15: Direct Scheme for Two Sinusoids

The results of an experiment are shown on Fig. 14. The signal at the microphone is plotted in dB as a function of time. The algorithm is not used until 0.5 sec. in the experiment, to show the amount of noise before compensation. The plot shows that the algorithm, once engaged, cancels the noise rapidly. Experiments were also performed using an algorithm designed for multiple harmonic components. The scheme is shown in Fig. 15 for the case of a fundamental and a third harmonic. The feedback compensation in the matrices $C_1(s)$ and $C_3(s)$ is similar to that used for a single component (including the G^{-1} matrix and the linear filters). A specific feature of the scheme is that the estimated frequencies of the two components of the control signal are tied together by a factor of 3, rather than being allowed to converge independently. Details of this scheme, and experimental results, are available in [13].

5. Chaplin and Smith's Patent

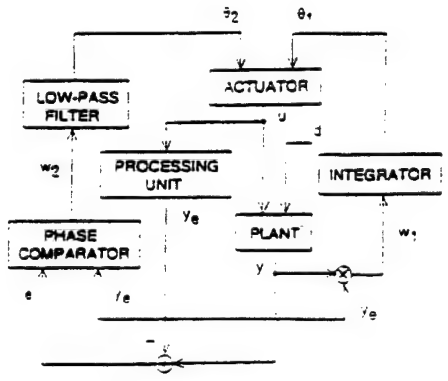

Figure 16: Chaplin and Smith's Scheme

Another scheme for periodic disturbance cancellation was proposed in the patent literature by Chaplin & Smith [14]. Fig. 16 shows a picture of the scheme. The right side of the figure shows the portion of the algorithm that determines the magnitude of the disturbance, while the left side is related to the determination of the frequency and phase of the disturbance. Although the frequency estimate is obtained in a similar way as in the modified indirect scheme, we view this concept as closer to the direct approach, because frequency and phase estimation are combined together.

The patent is not explicit regarding many important details, so that significant difficulties must be overcome to implement the concept and a large number of schemes could be obtained. A specific difficulty towards implementing the concept is that it requires a phase comparator whose output is zero when the two input signals are in phase. Typical phase comparators produce zero outputs for signals that are in quadrature. The lack of guidelines for the selection of the gains and of the filter parameters also have the result that naive attempts at implementing the scheme easily produce unstable closed-loop systems.

Some answers to these questions were provided

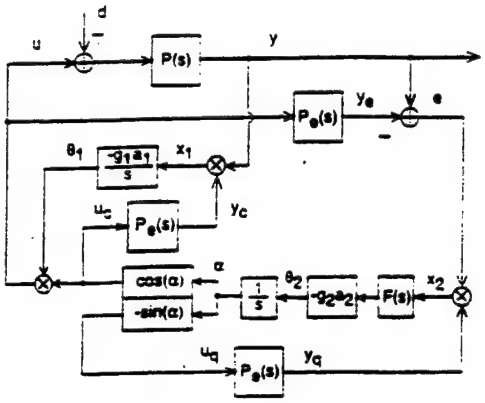

Figure 17: Modified Chaplin & Smith Scheme

in [15], and a practical scheme is shown in Fig. 17. An undesirable tendency for convergence of the original scheme towards a condition with $\theta_1 = 0$ was identified and resolved by picking some of the signals at the input of the plant, rather than at the output. A simple multiplier was implemented for the phase comparator, because a quadrature signal u_q was used instead of u. The low-pass filter was designed as a lead/lag filter with integral action, as for the direct scheme considered earlier. An analysis similar to the one used for that scheme yielded a successful design for the adaptive algorithm. After numerous adjustments made to the original concept, the performance of the scheme was found to be comparable to that of the direct scheme.

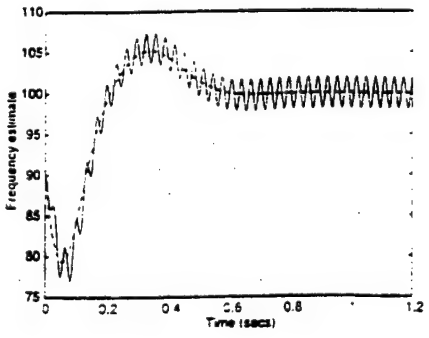

Figure 18: Frequency Estimate for the Modified Chaplin & Smith Scheme

Despite the similarities, however, there exists an interesting difference between the direct scheme and the modified scheme based on Chaplin & Smith's patent. Specifically, the signal x_2 converges to zero under ideal conditions in Fig. 10, because the output y converges to zero. On the other hand, the signal x_2 of Fig. 17 does not converge to zero, because the scheme uses a signal that is similar to the one used for the modified indirect scheme (interestingly, the use of such a signal is necessary for the indirect approach, but not for the direct approach). As a result, the frequency estimate θ_2 for the modified Chaplin & Smith scheme exhibits persistent fluctuations, shown in Fig. 18. These fluctuations do not necessarily lead to poor performance, and may be reduced by filtering. However, they are not present for the parameter of the direct scheme, shown in Fig. 13.

6. Conclusions

Three approaches were discussed for the feedback control of narrowband disturbances of unknown frequency. The indirect approach and the direct approach were found to be the most suitable for active noise control problems. The indirect approach was more flexible, but a more complete analysis was available for the direct approach. The algorithms based on the adaptive implementation of the internal model principle did not seem well-suited to active noise control problems in their current form. The scheme based on Chaplin & Smith patent was found to work well, provided that a significant number of adjustments were made to the original concept. The scheme could be viewed as a special case of the direct approach.

The paper emphasized the design of single-channel algorithms for pure sinusoidal disturbances. The extension of the schemes to multiple harmonic components is relatively straightforward, although pratical design may be harder. The extension to multi-channel systems is possible, but complex and an area of current research.

Acknowledgements

This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the pa-

per does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

References

1. S. Kuo, D. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, John Wiley & Sons, New York, NY, 1996.
2. K.S. Narendra, A. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
3. G. Feng, M. Palaniswamy, "Unified Treatment of Internal Model Principle Based Adaptive Control Algorithm," *Int. J. Control*, vol. 54, no. 4, pp. 883-901, 1991.
4. G. Feng, M. Palaniswamy, "A Stable Adaptive Implementation of the Internal Model Principle," *IEEE Trans. on Automatic Control*, vol. 37, no. 8, pp. 1220-1225, 1992.
5. M. Bodson, S. Douglas, "Narrowband Disturbance Rejection using Adaptive Feedback Algorithms," *Proc. of the SPIE Symposium on Smart Structures and Materials*, San Diego, CA, vol. 3039, pp. 45-56, 1997.
6. M. Bodson, A. Sacks, P. Khosla, "Harmonic Generation in Adaptive Feedforward Cancellation Schemes," *IEEE Trans. on Automatic Control*, vol. 39, no. 9, pp. 1939-1944, 1994.
7. A. Sacks, M. Bodson, P. Khosla, "Experimental Results of Adaptive Periodic Disturbance Cancellation in a High Performance Magnetic Disk Drive," *ASME Journal of Dynamics Systems, Measurement, and Control*, vol. 118, pp. 416-424, 1996.
8. M. Bodson, S. Douglas, "Adaptive Algorithms for the Rejection of Sinusoidal Disturbances with Unknown Frequency," *Automatica*, vol. 33, no. 12, pp. 2213-2221, 1997.
9. P.A. Regalia, "An Improved Lattice-Based Adaptive IIR Notch Filter," *IEEE Trans. on Signal Processing*, vol. 39, no. 9, pp. 2124-2128, 1991.
10. M. Bodson, S. Douglas, "Rejection of Disturbances with a Large Sinusoidal Component of Unknown Frequency," *Proc. of the SPIE Symposium on Smart Structures and Materials*, San Diego, CA, vol. 2715, pp. 64-75, 1996.
11. S. Sastry, M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Prentice Hall, Englewood Cliffs, NJ, 1989.
12. M. Bodson, "Cancellation of Sinusoidal Disturbances with Unknown Frequency," *Proc. of the Ninth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, pp. 174-179, 1996.
13. M. Bodson, J. Jensen, S. Douglas, "Active Noise Control for Periodic Disturbances," *Proc. of the American Control Conference*, Philadelphia, PA, June 24-26, 1998.
14. G. Chaplin, R. Smith, "Method of and Apparatus for Cancelling Vibration from a Source of Repetitive Vibrations," U.S. Patent 4,566,118, 1986.
15. M. Bodson, "A Discussion of Chaplin & Smith's Patent for the Cancellation of Repetitive Vibrations," submitted for presentation at the *IEEE Conference on Decision and Control*, Tampa, FL, Dec. 16-18, 1998.

Active Noise Control for Periodic Disturbances

Marc Bodson, Jonathan S. Jensen and Scott C. Douglas *
Department of Electrical Engineering, University of Utah
Salt Lake City, UT 84112, U.S.A.

Abstract: This paper proposes an active noise control algorithm for periodic disturbances of unknown frequency. The algorithm is appropriate for the feedback case in which a single error microphone is used. A previously-proposed algorithm for the rejection of sinusoidal noise sources is extended for the cancellation of multiple harmonics. Unlike many other approaches, the estimates of the frequencies of the separate harmonics are tied together within the algorithm to account for the integer multiplicative relations between them. The dynamic behavior of the closed-loop system is analyzed using an approximation that is shown, in simulations, to provide an accurate representation of the system's behavior. Experimental results on an active noise control testbed demonstrate the success of the method in a practical environment.

1. Introduction

The problem of active noise control is considered, as shown in Fig. 1. A microphone is used to measure the instantaneous noise level at some location to be made quiet. The signal is sampled and then processed by a digital signal processing system, and an anti-noise field is generated through a loudspeaker. The objective is to eliminate or significantly reduce the noise level at the microphone through destructive interference.

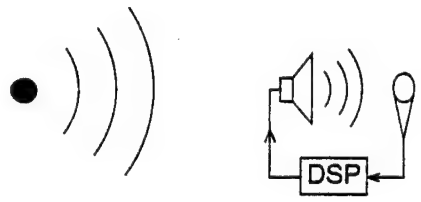

Figure 1: Active noise control (feedback scheme)

As shown in Fig. 1, the system configuration under consideration uses only one microphone. In other words, the situation is of a pure feedback nature, in contrast to the feedforward set-up that is often considered in such applications [1]. Moreover, the noise is assumed to be periodic in nature so that, from a control perspective, the problem is a classical rejection problem for periodic disturbances, except that the frequency of the disturbance is unknown and potentially time-varying. A limited number of approaches exist to address this problem. They include adaptive algorithms employing the internal model principle [2, 3, 4] and extensions of adaptive algorithms for disturbances of known frequency [5, 6].

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

This paper extends the algorithm of [6] to the more general case of periodic disturbances with multiple harmonics. A particular feature of the new algorithm is its use of the integer relationships between the harmonic components of the disturbance within the adaptive algorithm.

2. Adaptive Algorithm

2.1 Problem Statement

Assume that the transformation from the speaker to the microphone is a stable linear time-invariant system with transfer function $P(s)$ and that the effect of the noise source is additive. In the Laplace domain, the system can be modelled as

$$y(s) = P(s)(u(s) - d(s)), \tag{1}$$

where $y(s)$, $u(s)$ and $d(s)$ are the Laplace transforms of the microphone signal, of the speaker output, and of the equivalent noise signal at the speaker location, respectively. Alternatively, $u(t)$ may be viewed as the control input, $d(t)$ as the disturbance, and $y(t)$ as the plant output. The goal of the control system is to generate $u(t)$ such that $y(t) \to 0$ as $t \to \infty$. The objective would be achieved if $u(t) = d(t)$, but the disturbance $d(t)$ is not known or measured in any way except through its effect at the output of the system. It is assumed to be a periodic signal, so that

$$\begin{aligned} d(t) &= \sum_{k=1}^{n} d_k \cos(\alpha_{k,d}(t)), \\ \dot{\alpha}_{k,d}(t) &= k\omega_1. \end{aligned} \tag{2}$$

The parameters ω_1, d_k, and $\alpha_{k,d}(0)$ (for $k = 1, ..., n$) are unknown. The order of the highest harmonic, n, is assumed to be finite and known. Certain harmonics may also be specified to be absent, *i.e.*, certain values of d_k may be known to be small *a priori*. For simplicity of presentation, we will consider the case where the fundamental and the third harmonic are present (only d_1 and d_3 are nonzero).

2.2 Adaptive Algorithm

The structure of the proposed scheme is shown in Fig. 2. The signal u_1 nominally cancels the fundamental (at frequency ω_1), while the signal u_3 cancels the third harmonic. The parameters α_1 and α_3 are the estimates of the angles of the two components of the disturbance. The parameters θ_{11} and θ_{31} are the estimates of the magnitudes of the two sinusoids, and θ_{12} is the estimate of the frequency of the fundamental. The value of θ_{12} is integrated to obtain the angle α_1. For the third harmonic, θ_{32} is not the frequency but rather is the relative phase of the signal. The algorithm uses the assumption that the second sinusoid is a third harmonic of the fundamental

0-7803-4530-4/98 $10.00 © 1998 AACC

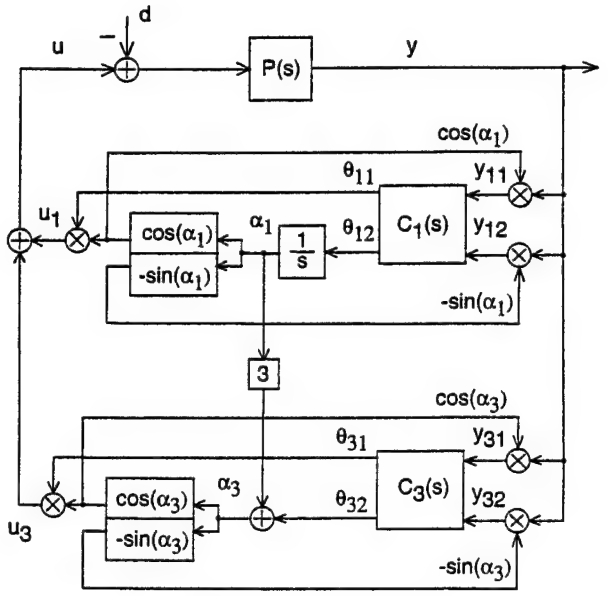

Figure 2: Adaptive Algorithm

by letting the angle α_3 be equal to the sum of three times the angle of the fundamental and of the relative phase θ_{32}.

The equations for the control algorithm are thus

$$\begin{aligned} u &= \theta_{11}\cos(\alpha_1) + \theta_{31}\cos(\alpha_3) \\ \dot{\alpha}_1 &= \theta_{12}, \quad \alpha_3 = 3\cdot\alpha_1 + \theta_{32} \\ y_{11} &= y\cos(\alpha_1), \quad y_{12} = -y\sin(\alpha_1) \\ y_{31} &= y\cos(\alpha_3), \quad y_{32} = -y\sin(\alpha_3). \end{aligned} \tag{3}$$

The parameters have nominal values $\theta_{11}^* = d_1$, $\theta_{12}^* = \omega_1$, $\theta_{31}^* = d_3$, and $\theta_{32}^* = \alpha_{3,d}(0) - 3\alpha_{1,d}(0)$. For these values, and for $\alpha_1(t) - \omega_1 t = \alpha_{1,d}(0)$, the output converges to zero.

The transfer function matrix $C_1(s)$ relates the signals y_{11} and y_{12} to the parameters θ_{11} and θ_{12}, respectively. Similarly, the transfer function matrix $C_3(s)$ relates the signals y_{31} and y_{32} to the parameters θ_{31} and θ_{32}. The matrices $C_1(s)$ and $C_3(s)$ are the products of constant matrices with diagonal transfer function matrices. They are defined as follows. Consider the real and imaginary parts of the plant frequency response at the two frequencies of interest, given by

$$\begin{aligned} P_{R,1} &= Re[P(j\omega_1)], \quad P_{R,3} = Re[P(3j\omega_1)], \\ P_{I,1} &= Im[P(j\omega_1)], \quad P_{I,3} = Im[P(3j\omega_1)], \end{aligned} \tag{4}$$

and define the matrices

$$G_1 = \begin{pmatrix} P_{R,1} & -P_{I,1} \\ P_{I,1} & P_{R,1} \end{pmatrix}, G_3 = \begin{pmatrix} P_{R,3} & -P_{I,3} \\ P_{I,3} & P_{R,3} \end{pmatrix}. \tag{5}$$

Variables x_{11}, x_{12}, x_{31}, and x_{32} are defined as

$$\begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix} = G_1^{-1}\begin{pmatrix} y_{11} \\ y_{12} \end{pmatrix}, \begin{pmatrix} x_{31} \\ x_{32} \end{pmatrix} = G_3^{-1}\begin{pmatrix} y_{31} \\ y_{32} \end{pmatrix}, \tag{6}$$

and the algorithm parameters are given by

$$\begin{aligned} \dot{\theta}_{11} &= -2g_1x_{11}, \quad \dot{\theta}_{12} = -2g_2x_{12f}/d_{1e}, \\ \dot{\theta}_{31} &= -2g_3x_{31}, \quad \dot{\theta}_{32} = -2g_3x_{32}/d_{3e}. \end{aligned} \tag{7}$$

The update laws are defined through integral relationships to guarantee zero steady-state errors. The control law for θ_{12} is slightly different from the others, with the signal x_{12} filtered so that

$$x_{12f}(s) = F(s)x_{12}(s), \quad F(s) = \frac{s+a}{s+b}. \tag{8}$$

The compensation filter is necessary to ensure the stability of the closed-loop system. The constants a, b, g_1, g_2, and g_3 will be adjusted to obtain satisfactory performance. The parameters d_{1e} and d_{3e} are estimates of d_1 and d_3 (see section 3.2).

To extend the algorithm for arbitrary harmonics, additional paths similar to that for the third harmonic in Fig. 2 may be added. The multiplying factor of 3, the matrix G_3, and the estimate d_{3e} are the only elements that need to be changed. Note that, in the proposed implementation, frequency estimation is provided through the first harmonic. This choice is not essential, and frequency estimation based on another harmonic is possible.

3. Stability Analysis and Control Design

3.1 Stability Analysis

Our analysis of the system is based on a fundamental fact, whose proof is reminiscent of derivations found in the study of frequency-modulation communication systems [7] and of averaging methods applied to adaptive systems [8]. The following conditions are assumed:

- the values of θ_{11}, θ_{12}, θ_{31}, and θ_{32} vary sufficiently slowly that the response of the plant to the signal $u(t)$ may be approximated by the steady-state output of the plant for the two sinusoidal signals with frequencies θ_{12} and $3\theta_{12}$.
- the instantaneous frequency θ_{12} is close to ω_1, so that $P(j\theta_2)$ may be replaced by $P(j\omega_1)$ and $P(3j\theta_2)$ may be replaced by $P(3j\omega_1)$.

Basic Fact: Considering low-frequency components only, the signals $x_{11}(t)$, $x_{12}(t)$, $x_{31}(t)$, and $x_{32}(t)$ are approximately given by

$$\begin{pmatrix} x_{i1}(t) \\ x_{i2}(t) \end{pmatrix} = \frac{1}{2}\begin{pmatrix} \theta_{i1}(t) - \theta_{i1}^*\cos(\alpha_i(t) - \alpha_{i,d}(t)) \\ \theta_{i1}^*\sin(\alpha_i(t) - \alpha_{i,d}(t)) \end{pmatrix} \tag{9}$$

for $i = \{1, 3\}$, with

$$\begin{aligned} \alpha_1(t) - \alpha_{1,d}(t) &= \int_0^t (\theta_{12}(\sigma) - \theta_{12}^*)d\sigma + \alpha_1(0) - \alpha_{1,d}(0), \\ \alpha_3(t) - \alpha_{3,d}(t) &= 3(\alpha_1(t) - \alpha_{1,d}(t)) + \theta_{32}(t) - \theta_{32}^*. \end{aligned} \tag{10}$$

The proof follows similar steps as in the proof in [6] and is omitted. The elimination of the high-frequency components within the system can be achieved through low-pass filtering of the signals. However, the signals are filtered within the compensators $C_1(s)$ and $C_3(s)$, and although the filtering is not ideal, simulations show that it is sufficient for the satisfactory operation of the system.

3.2 Compensator Design

Although the equations are nonlinear, a linear system is obtained if the parameters are close to their nominal

values and the phase error $\delta\alpha_1(t) = \alpha_1(t) - \alpha_{1,d}(t)$ is small. The linearized system is

$$\begin{pmatrix} x_{11}(t) \\ x_{12}(t) \end{pmatrix} = \frac{1}{2}\begin{pmatrix} \theta_{11}(t) - \theta_{11}^* \\ \theta_{11}^* \delta\alpha_1(t) \end{pmatrix},$$
$$\begin{pmatrix} x_{31}(t) \\ x_{32}(t) \end{pmatrix} = \frac{1}{2}\begin{pmatrix} \theta_{31}(t) - \theta_{31}^* \\ \theta_{31}^* \left(3\delta\alpha_1(t) + \theta_{32}(t) - \theta_{32}^*\right) \end{pmatrix} \quad (11)$$

The linearized dynamics from the parameters θ_{11} and θ_{12} to the variables x_{11} and x_{12} are decoupled from one another and are not dependent on the dynamics of the variables associated with the third harmonic. In closed-loop, the dynamics of θ_{11} are those of a first-order system with a pole at $s = -g_1$. For θ_{12}, the closed-loop poles are determined by the roots of $s^2(s+b) + g_2(s+a) = 0$, if $d_{1e} = d_1 = \theta_{11}^*$. Otherwise, g_2 is replaced by $g_2 d_1/d_{1e}$. Stability is guaranteed if $g_2 > 0$ and $b > a > 0$. For the variables associated to the third harmonic, the closed-loop dynamics are those of a first-order system with a pole at $s = -g_3$. For θ_{32}, the pole is at $s = -g_3 d_3/d_{3e}$ if d_{3e} is not equal to $d_3 = \theta_{31}^*$. The phase error $\delta\alpha_1(t)$ also appears as a disturbance on the equation for θ_{32}. The stability of the equation for θ_{12} ensures that this disturbance vanishes with time.

Because the magnitudes of the sinusoidal components d_1 and d_3 act as gains in the two transfer functions associated with phase locking, estimates of the parameters are used to ensure that the closed-loop poles are set at desirable values. However, the stability of the linear systems is not dependent upon the accuracy of these estimates. The algorithm also requires the knowledge of the matrices G_1 and G_3, which depend on ω_1. One may choose to set these matrices for a value of the frequency in the middle of the expected range of operation, or one may use the estimated frequency θ_{12} in real-time. Either way, knowledge of the frequency response of the plant in the frequency range of interest is required.

4. Simulation Results

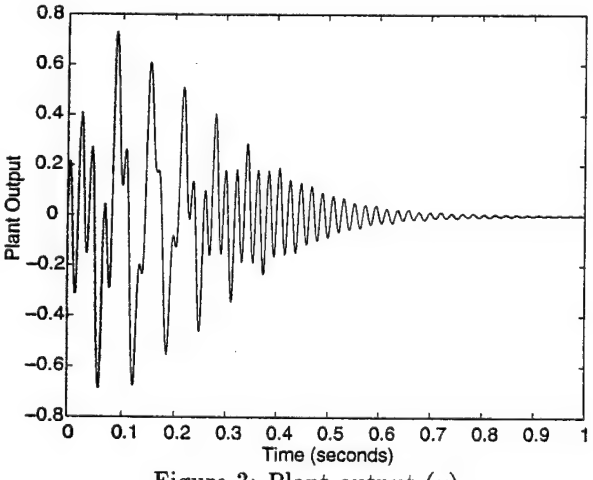

Figure 3: Plant output (y)

The performance of the algorithm is examined via simulation. We consider a situation in which $P(s) = 100/(s+100)$, $\omega_1 = 100$, $d_1 = 1$, and $d_3 = -1$. The parameters

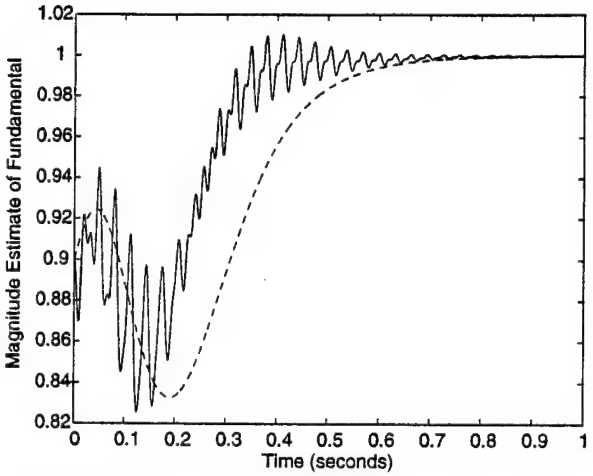

Figure 4: Magnitude of the fundamental (θ_{11})

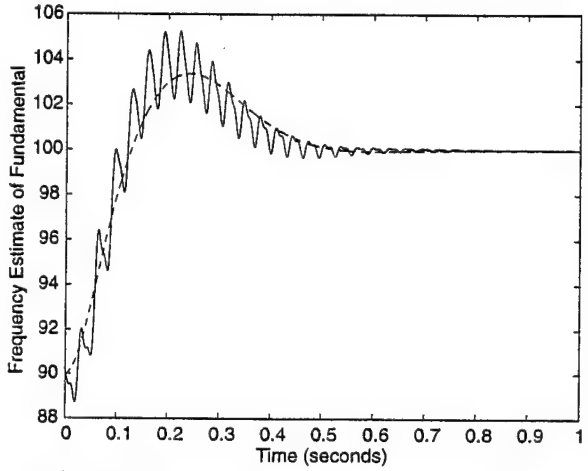

Figure 5: Frequency of the fundamental (θ_{12})

g_1 and g_3 are set to 10, leading to closed-loop poles for the first-order systems at -10 rad/s. The other parameters are set to $g_2 = 400$, $a = 5$, and $b = 30$, leading to closed-loop poles for the frequency control loop located at -10 rad/s and $-10 \pm j10$ rad/s. The initial states of the parameters are zero, except for $\theta_{11}(0) = 0.9$, $\theta_{12}(0) = 90$ rad/s.

Fig. 3 shows the output of the plant, which is found to decrease to negligible values in less than a second. The transient behavior of the magnitude estimate θ_{11} is shown in Fig. 4, where the solid line is the parameter response, and the dashed line is the response predicted from a simulation of an approximate system. The approximate system is composed of the control law (7), (8), and the nonlinear approximation (9), (10). The frequency estimate θ_{12} is shown in Fig. 5. The matches between the actual and approximate behaviors is good, and the degree of match is similar to those for other adaptive systems [8].

The magnitude estimate for the third harmonic θ_{31} is shown in Fig. 6 and the phase estimate θ_{32} in Fig. 7. Note that the magnitude θ_{31}^* was set to -1, and the estimate of the magnitude converged to 1, with the estimate of the phase converging to $-\pi$, or -180^0. Again, the approximations are very good, and although the nonlin-

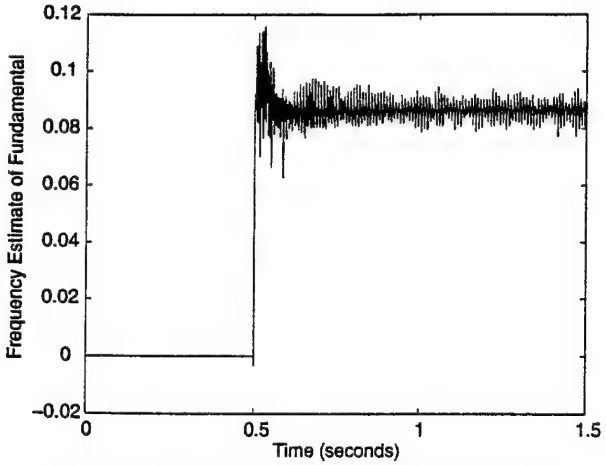

Figure 10: Frequency of the fundamental (θ_{12})

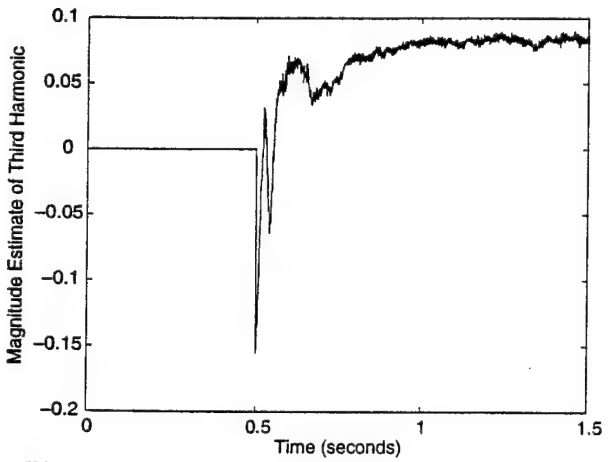

Figure 11: Magnitude of the third harmonic (θ_{31})

estimate θ_{12}). The units of the frequency estimate are given in radians/sample; *i.e*, $2\pi \cdot 110/8000 = 0.086$. The initial frequency estimate corresponded to 115 Hz. Other parameters were initially set to zero. Shown in Fig. 11 and 12 are the parameters related to the third harmonic (magnitude estimate θ_{31} and phase estimate θ_{32} (in rad)). It was found that the frequency of the disturbance could vary over a wide range, once the algorithm had locked onto the disturbance frequency. Techniques from phase-locked loops could be used to expand the lock-in range [7] but were not implemented. Other experiments employing several harmonics as well as frequency estimation using high-order harmonics provided results comparable to those shown.

6. Conclusions

An adaptive algorithm is proposed for the rejection of periodic disturbances of unknown frequency. For simplicity, the algorithm is described for a noise consisting of a fundamental component and a third harmonic, although the method is easily extended to noises with an arbitrary number of harmonics. As in other solutions to this control problem, the closed-loop system is a complex

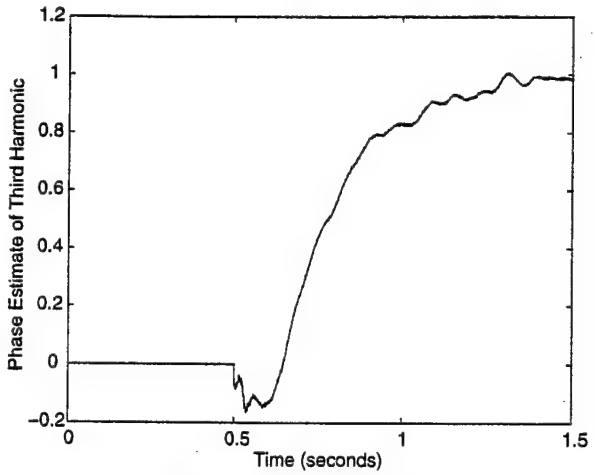

Figure 12: Phase of the third harmonic (θ_{32})

nonlinear time-varying system. However, we showed that an approximate nonlinear time-invariant system provided a very accurate representation of the dynamic behavior of the system. A further approximation of the system through linearization was useful for the selection of the design parameters. The algorithm was tested experimentally on an active noise control system at the University of Utah to demonstrate the success of the method in a practical environment.

7. References

[1] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, New York, Wiley, 1996.

[2] K.S. Narendra and A. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] G. Feng and M. Palaniswamy, "Unified Treatment of Internal Model Principle Based Adaptive Control Algorithm," *Int. J. Control*, vol. 54, no. 4, pp. 883-901, 1991.

[4] G. Feng and M. Palaniswamy, "A Stable Adaptive Implementation of the Internal Model Principle," *IEEE Trans. on Automatic Control*, vol. 37, no. 8, pp. 1220-1225, 1992.

[5] G.B.B. Chaplin and R.A. Smith, "Method of and Apparatus for Cancelling Vibrations from a Source of Repetitive Vibrations," U.S. Patent 4,566,118, Jan. 21, 1986.

[6] M. Bodson and S.C. Douglas, "Adaptive Algorithms for the Rejection of Periodic Disturbances with Unknown Frequency," *Automatica*, vol. 33, no. 12, pp. 2213-2221, 1997.

[7] J.R. Smith, *Modern Communications Circuits*, Mc Graw-Hill, New York, NY, 1997.

[8] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

ear effects are significant, the design based on the linear approximation is adequate to obtain convergence of the system.

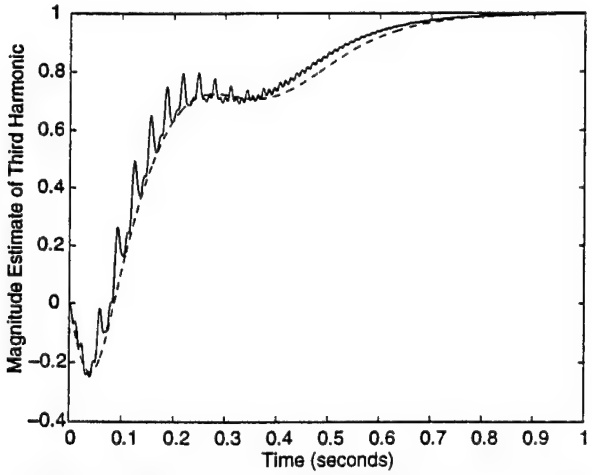

Figure 6: Magnitude of the third harmonic (θ_{31})

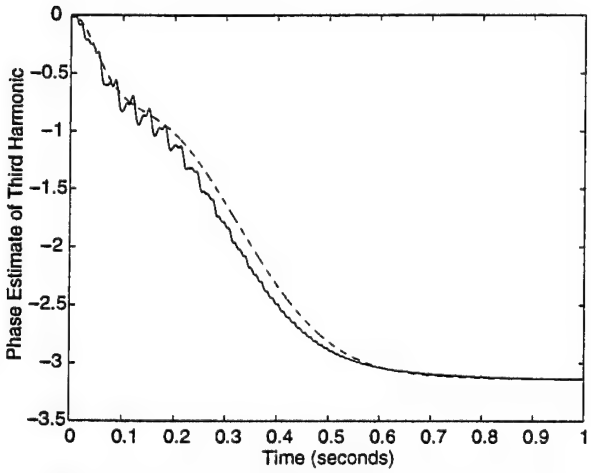

Figure 7: Phase of the third harmonic (θ_{32})

5. Experimental Results

The scheme was implemented on an experimental active noise control system developed at the University of Utah. The algorithm was coded in assembly language on a Motorola DSP96002 32-bit floating-point digital signal processor, The sampling rate was set at 8 kHz. A single bookshelf speaker with a 4-inch low-frequency driver, located approximately 2 ft away from the error microphone, generated a periodic signal constituting the noise source. The microphone signal was passed through an anti-aliasing filter and sampled by a self-calibrating 16-bit analog-to-digital converter before being sent to the DSP system. The controller output signal was sent to a noise cancelling speaker placed approximately 1 ft away from the microphone. Only a single error sensing microphone signal was used.

The frequency response of the plant was determined during a rapid calibration phase in which pure sinusoidal tones were applied to the noise cancellation speaker, and the responses were measured by the microphone signal. The real and imaginary parts of the frequency response

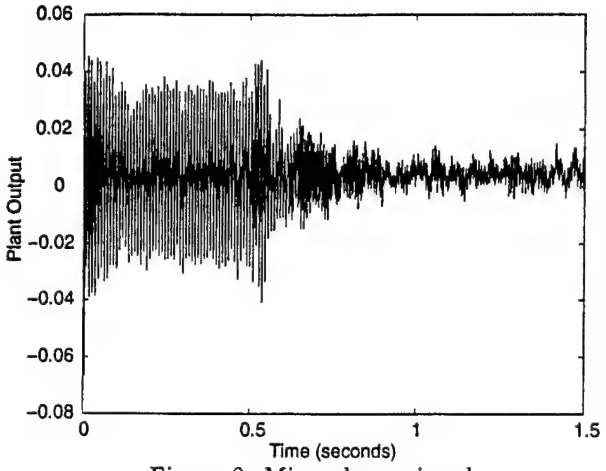

Figure 8: Microphone signal

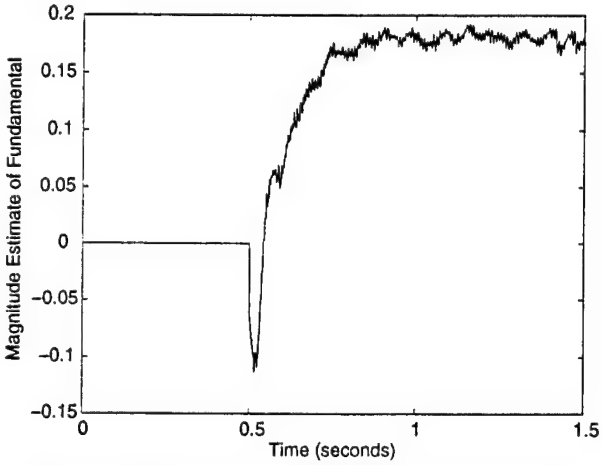

Figure 9: Magnitude of the fundamental (θ_{11})

were obtained at 16 different frequencies, spaced logarithmically between 32.5 Hz and 1 kHz. In other experiments, the frequency response was calculated from a 50-tap finite impulse response model obtained with an adaptive identification algorithm and a white noise input. This identification procedure took longer. The matrices G_1 and G_3 were adjusted in real-time, based on the frequency estimate θ_{12}. Values from a look-up table were interpolated linearly as needed. Update of the matrices was performed every 8 samples. Because of the digital implementation, a discrete-time equivalent of the algorithm was implemented, with the z-domain poles placed in the vicinity of $z = 0.995$ for the parameters of the fundamental and $z = 0.99$ for the parameters of the third harmonic.

The results of one experiment are shown on Fig. 8, in which the signal at the error microphone is plotted as a function of time. The algorithm is not engaged until 0.5 sec. in the experiment, so that the amount of noise before compensation can be judged. The frequency of the fundamental is 110 Hz. The plot shows that the algorithm, once engaged, cancels the noise within a fraction of a second. The adaptive parameters for the fundamental are shown in Fig. 9 (magnitude estimate θ_{11}) and Fig. 10 (frequency

A Discussion of Chaplin and Smith's Patent for the Cancellation of Repetitive Vibrations

Marc Bodson

***Abstract*—This paper considers a method proposed by Chaplin and Smith for the cancellation of repetitive vibrations. Direct implementation of the concepts of the patent is not straightforward because of the lack of precise information regarding certain components, and because of the absence of guidelines for the design of a system with satisfactory stability and performance properties. This paper proposes a specific algorithm based on the concepts of the patent, and provides an approximate analysis which is found useful to obtain a closed-loop system with predictable dynamic characteristics. Simulations demonstrate the validity of the approximation and the success of the design. A tendency of the algorithm for convergence toward undesirable operating states is identified, and an improved version is derived. The algorithm is also compared to another algorithm proposed for the same purposes.**

***Index Terms*—Active noise control, active vibration control, adaptive control, disturbance rejection.**

I. Introduction

The active control of noise and vibration is useful as an alternative to expensive passive damping methods. In many applications, the algorithms are what are called *adaptive feedforward control* algorithms. A measurement is taken from the source of the disturbance: for example, a microphone or accelerometer placed on or close to the source. The control algorithm is then inherently of a feedforward nature, although feedback is used for adaptation. When feedforward measurements are not available, one must rely on feedback algorithms to provide for compensation. Such algorithms are usually more difficult to design. Among the problems for which adaptive feedback compensation may be considered are those where the disturbance signal is periodic. This case occurs when the source of the disturbance is a rotating machine. The disturbance rejection problem for periodic disturbances of unknown frequency, using feedback information only, is called the *narrow-band adaptive feedback case* [1]. For simplicity, we will assume that the disturbance has a single harmonic component, i.e., that it is sinusoidal. More complex algorithms can be built to cancel multiple harmonics using the approaches developed for a single component.

Few algorithms are specifically targeted to the narrow-band adaptive feedback case. An approach is based on the adaptive implementation of the internal model principle [2], [3]. These algorithms have the advantage of providing adaptation to unknown plant parameters, as well as to unknown disturbance parameters. However, they have delicate stability properties, and are not well suited to active noise and vibration control problems [4]. Another approach consists of extending the capabilities of existing adaptive algorithms. An algorithm, proposed in [5], combines frequency estimation with an adaptive algorithm for the rejection of disturbances with known frequency. Another algorithm combines the estimation of the magnitude, frequency, and phase in a single control loop. Chaplin and Smith propose their own method in [6]. While many details are missing that are needed for the implementation of the concepts, we propose our own interpretations and extensions of the original concepts to obtain an algorithm that is shown, in simulations, to achieve the desired results. To make the design possible, an approximate analysis method is presented that allows one to accurately predict the dynamic characteristics of the closed-loop system. Interestingly, the analysis method is the same as was used in [5], and is similar to the averaging methods applied to adaptive systems in [7], and to the approximations made in the analysis of phase-locked loops [8].

Manuscript received September 9, 1998. Recommended by Associate Editor, M. M. Polycarpou. This work was supported by the U.S. Army Research Office under Grant DAAH04-96-1-0085.

The author is with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA (e-mail: bodson@ee.utah.edu).

Publisher Item Identifier S 0018-9286(99)08595-5.

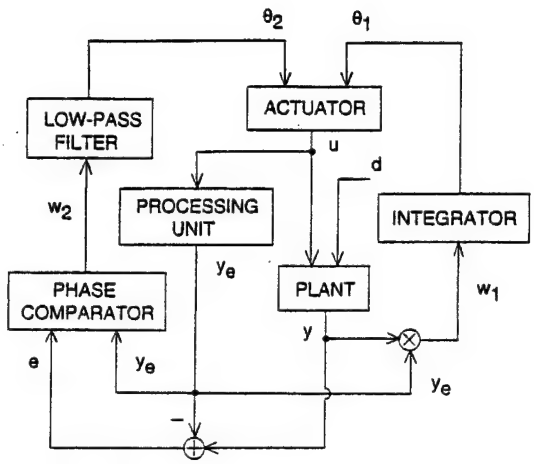

Fig. 1. Chaplin and Smith's method for canceling repetitive vibrations.

II. Chaplin and Smith's Method

Chaplin and Smith begin the discussion of their patent [6] by presenting a scheme for the cancellation of periodic vibrations relying on measurements taken from the machine that is the source of the vibration. Then, an approach is proposed to remove the requirement of the measurement, and several feedback schemes are proposed, with each scheme building on the previous one. The descriptions leave many details unspecified, so that a wide variety of algorithms could be obtained through application of the concepts of the patent. Our study applies to the last scheme, which is the most specific.

Fig. 1 shows the scheme under consideration. It is essentially [6, Fig. 5]. The only differences are that the plant has been identified on the figure, and the signals have been labeled for the purpose of the discussion. The plant has control input u and output y. It is perturbed by a disturbance d. The effects of u and d are assumed to be additive, so that the disturbance may be cancelled by the control signal. The disturbance is assumed to be a sinusoidal signal with unknown magnitude, frequency, and phase. An actuator is shown, whose output is a sinusoidal signal with magnitude and frequency specified by the inputs θ_1 and θ_2, respectively.

In the scheme, the frequency of the disturbance is estimated using a phase-locked loop. As pointed out by Chaplin and Smith, a difficulty could arise if the output of the plant y were used for frequency estimation. The reason is that the cancellation scheme is designed to eliminate the signal y on which frequency estimation would be based. To avoid the problem, an estimated output y_e is reconstructed using a so-called *processing unit*. The signal y_e is then subtracted from the output y to produce an error signal e which, in theory, is equal to the

component of the output due to the disturbance (i.e., the output that would be observed if the control input were zero).

A phase comparator produces a signal w_2 proportional to the phase error between y_e and e, in a configuration that is typical of phase-locked loops. Note that the difference in phase between y_e and e is equal, in theory, to the difference of the phase between the signal u and its desired value. Therefore, the phase error signal w_2 may be used to adjust the frequency θ_2 and, indirectly, the phase of the actuator signal. A low-pass filter is inserted to eliminate the high frequencies typically associated with phase comparators.

The magnitude of the control signal θ_1 is adjusted using a feedback loop appearing on the right of Fig. 1. The plant output y and the estimated plant output y_e are multiplied, and the resulting signal w_1 is filtered by an integrator to produce the magnitude signal for the actuator. The idea is that, if the phase of the control signal is correct, the product $y \cdot y_e$ is positive on the average if θ_1 is less than its desired value, and negative otherwise.

Note that the patent proposes the scheme of Fig. 1 for specific application to vibration control, and for the case where a special actuator is used to generate a sinusoidal force. The magnitude and frequency of the sinusoidal excitation are the input signals of the actuator. However, the scheme may be applied to a broad range of problems where periodic disturbances of unknown frequency must be rejected. To use an arbitrary actuator, it is only necessary to generate, in analog or digital form, a sinusoidal wave with prescribed magnitude and frequency.

On the other hand, there are two limitations to the patent. The first is that it does not provide clear guidelines for a successful implementation. Experimentation with the scheme shows that it is not difficult to obtain an unstable feedback system. Because the closed-loop system is nonlinear and time varying, analysis is not straightforward. Our analysis, discussed later in the paper, addresses this issue and, further, shows how to select the low-pass filter to ensure the stability of the closed-loop system.

Another difficulty with the direct implementation of the method is that the scheme requires a phase comparator whose output is zero for zero phase error. Phase comparators normally produce a signal whose output is zero when the two input signals are in quadrature (90° out of phase). The problem could be resolved by shifting one of the signals by 90°. However, the design of a 90° phase shifter is difficult in applications where the frequency of the disturbance varies over a wide range. We propose a simple solution to this problem, and derive a concrete algorithm.

III. Proposed Implementation of the Method

Fig. 2 shows our proposed implementation of the method. Fig. 1 has been reorganized in a diagram whose structure is closer to that of conventional control systems. The additive effect of the disturbance is also shown explicitly. The plant is assumed to be a linear time-invariant system with transfer function $P(s)$. The processing unit of Fig. 1 is replaced by a transfer function $P_e(s)$, which is an estimate of $P(s)$. The estimate of the magnitude θ_1 is obtained as in the original figure, except that gains g_1 and a_1 were introduced as design parameters. The separation of the gain into two gains g_1 and a_1 is made for convenience, for reasons to become clearer later.

For the estimate of the frequency θ_2, a multiplier is used as a phase comparator (it is the simplest phase comparator, and by no means the only choice available). Rather than shifting the phase of the signal y_e by 90°, a quadrature signal y_q is obtained by filtering a quadrature input signal u_q by the transfer function $P_e(s)$. The signal u_q is obtained in parallel with the signal u by using a sine function instead of a cosine function. The filter $F(s)$ may serve the purpose of the low-pass filter shown in Fig. 1. However, we will design it as

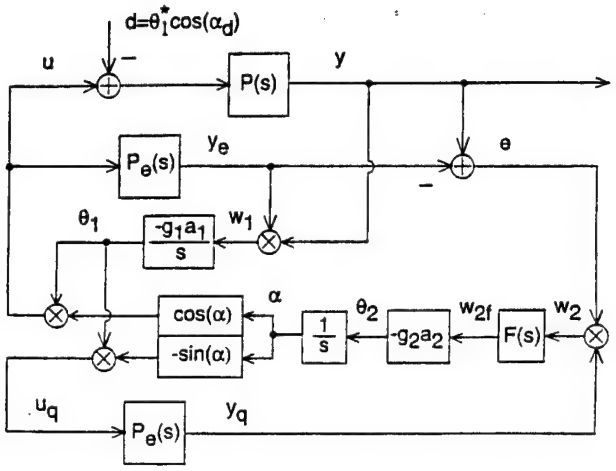

Fig. 2. Implementation of Chaplin and Smith's method.

a compensation filter to ensure stability and performance. Two gains g_2 and a_2 are again inserted as design parameters.

The equation for the plant is, in the Laplace domain, $y(s) = P(s)(u(s) - d(s))$, where $y(s)$, $u(s)$, and $d(s)$ are the Laplace transforms of the plant output, the plant input, and the disturbance signal, respectively. The disturbance $d(t)$ is assumed to be a sinusoid with fixed magnitude θ_1^* and fixed frequency ω_1, so that

$$d(t) = \theta_1^* \cos(\alpha_d(t)), \qquad \dot{\alpha}_d(t) = \omega_1. \tag{1}$$

The parameters θ_1^*, ω_1, and $\alpha_d(0)$ are all unknown.

The equations for the control algorithm are

$$u(t) = \theta_1(t) \cos(\alpha(t)), \qquad \dot{\alpha}(t) = \theta_2(t), \tag{2}$$

while the equations for the parameters θ_1, θ_2 and other variables are

$$\begin{aligned} \dot{\theta}_1(t) &= -g_1 a_1 w_1(t), & \dot{\theta}_2(t) &= -g_2 a_2 w_{2f}(t) \\ w_1(t) &= y(t) \cdot y_e(t), & w_2(t) &= (y(t) - y_e(t)) \cdot y_q(t) \\ u_q(t) &= -\theta_1(t) \sin(\alpha(t)). \end{aligned} \tag{3}$$

The remaining variables are obtained through linear time-invariant filters

$$\begin{aligned} y_e(s) &= P_e(s)u(s) \\ y_q(s) &= P_e(s)u_q(s) \\ w_{2f}(s) &= F(s)w_2(s). \end{aligned} \tag{4}$$

IV. Approximate Analysis

The analysis method followed here was used earlier for a closely related scheme [5]. It is similar to derivations found in the analysis of frequency-modulation communication systems [8], and to averaging methods used for adaptive systems [7]. We consider the following assumptions.

Assumptions:

1) The plant is stable, and $P_e(s) = P(s)$ (this equality is not required for the analysis, but is assumed for simplicity).
2) The signals $\theta_1(t)$ and $\theta_2(t)$ vary sufficiently slowly that the response of the plant to the signal $u(t)$ can be approximated by the steady-state output of the plant for a sinusoidal input with fixed magnitude θ_1 and fixed frequency θ_2.
3) The signals $y_e(t)$ and $y_q(t)$ are assumed to be approximately given by the steady-state outputs of the transfer function $P_e(s)$ with inputs u and u_q, respectively.

4) The instantaneous frequency θ_2 is close to ω_1, so that $P(j\theta_2)$ can be approximated by $P(j\omega_1)$.

Next, denoting $P_R = \mathrm{Re}[P(j\omega_1)]$, $P_I = \mathrm{Im}[P(j\omega_1)]$, and $P_M^2 = P_R^2 + P_I^2$, we obtain the following fact.

Fact: Under the assumptions, and considering low-frequency components only, the two signals $w_1(t)$ and $w_2(t)$ are approximately given by

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} = \tfrac{1}{2}\theta_1(t)P_M^2 \begin{bmatrix} \theta_1(t) - \theta_1^* \cos(\delta\alpha(t)) \\ \theta_1^* \sin(\delta\alpha(t)) \end{bmatrix} \tag{5}$$

where $\delta\alpha(t) = \alpha(t) - \alpha_d(t)$.

Proof: The proof is omitted for brevity.

Comments: The elimination of the high-frequency components can be achieved by low-pass filtering of the signals $w_1(t)$ and $w_2(t)$, as is commonly done in phase-locked loops. Fig. 2 shows that, in both paths, signals are filtered by an integrator. Although more effective filters may be used, their filtering benefits would have to be traded off with the delays that would be imposed. We assume here that no additional filter is applied to eliminate the high-frequency components.

V. Control System Design

Equation (5) can be viewed as an alternative description of the plant, with two inputs θ_1 and α, and two outputs w_1 and w_2. Although the description is nonlinear, a linearized system may be obtained by assuming that the magnitude error $\theta_1 - \theta_1^*$ and that the phase error $\delta\alpha = \alpha - \alpha_d$ are small. Specifically, the linearized system is

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} = \tfrac{1}{2}\theta_1^* P_M^2 \begin{bmatrix} \theta_1(t) - \theta_1^* \\ \theta_1^* \delta\alpha(t) \end{bmatrix} \tag{6}$$

with $\delta\dot{\alpha}(t) = \theta_2(t)$. The linearized dynamics of the system from the parameters θ_1 and θ_2 to the variables w_1 and w_2 are decoupled from one another, and are those of a simple gain of $(1/2)\theta_1^* P_M^2$ and of an integrator with a gain $(1/2)(\theta_1^*)^2 P_M^2$, respectively. To compensate for the effect of the two gains, the constants a_1 and a_2 of the control algorithm are chosen to be the inverses of the two gains, i.e., $a_1 = 2/(\theta_1^* P_M^2)$, $a_2 = 2/((\theta_1^*)^2 P_M^2)$. A prior estimate of θ_1^* is needed to set the gains, in addition to an estimate of the frequency response. The estimates do not need to be precise, but sufficiently accurate for the stability of the closed-loop system to be maintained.

The control loops for θ_1 and θ_2 are adjusted to ensure stability of the nominal system and the rejection of the constant disturbances created by the unknown parameters. The dynamics of the closed-loop system for θ_1 are those of a first-order system with pole at $s = -g_1$. The parameter may be set to place the pole at a desirable location. The closed-loop system for the second system has dynamics determined by the roots of $s + g_2 F(s) = 0$. Because of the presence of a constant disturbance $\alpha_d(0)$, it is useful to include integral compensation, i.e., to let $F(s)$ have a pole $s = 0$. Such a choice that also guarantees stability is $F(s) = (s + a)/(s(s + b))$ with $g_2 > 0$ and $b > a > 0$. Again, the parameters may be chosen to place the roots of $s^2(s + b) + g_2(s + a) = 0$ at desirable locations.

VI. Simulation Results

We demonstrate the validity of the approach through simulations. We let $P(s) = P_*(s) = 100/(s + 100)$, $\omega_1 = 100$, and $d_1 = 1$. The parameter g_1 is set to 10, leading to a closed-loop pole for the first control loop (magnitude loop) at -10 rad/s. The other parameters are set to $g_2 = 400$, $a = 5$, and $b = 30$, leading to closed-loop poles for the second control loop (frequency and phase loop) at -10 and $-10 \pm j10$ rad/s. The initial states of all of the parameters are zero,

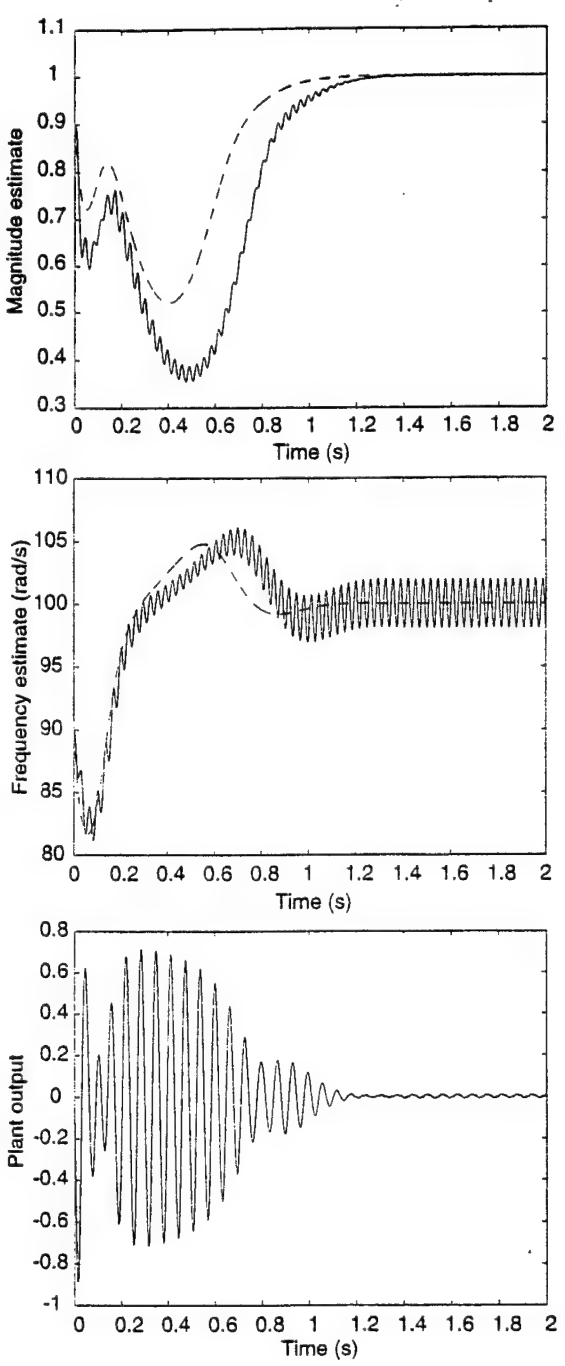

Fig. 3. Responses of the algorithm.

except for $\theta_1(0) = 0.9$, $\theta_2(0) = 90$ rad/s, and $\alpha(0) = 90°$. In other words, the system is initialized with a 10% error in magnitude and frequency, and a phase error away from the linear region.

Fig. 3 shows the responses of the system. From top to bottom, the magnitude estimate θ_1, the frequency estimate θ_2, and the output of the plant are shown. For the estimates, the dashed lines show the responses predicted by simulating the nonlinear system composed of the control law from w_1, w_2 to θ_1, θ_2 and the approximation (5). The output is found to decrease to small values in about 2 s, and the predictions using the approximate nonlinear system are good. The linearized approximate system was used for the design with success.

In the simulations, the system converged despite a 10% initial error in magnitude and frequency. However, an undesirable behavior was found to occur for an initial frequency error of opposite sign, and it is

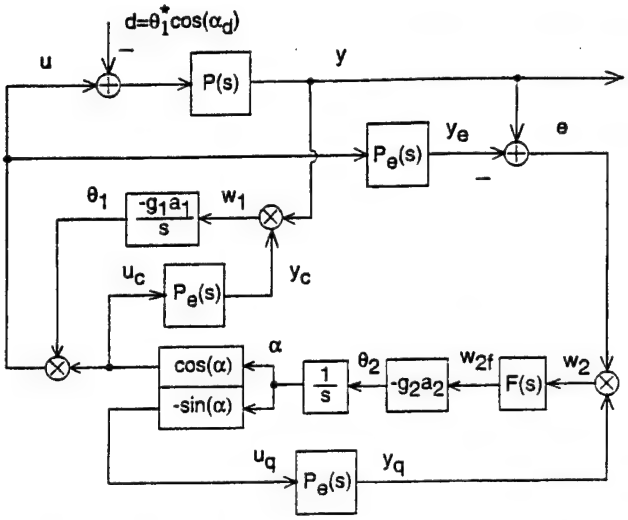

Fig. 4. Improved implementation of Chaplin and Smith's method.

discussed in the next section. In any case, one should expect a finite region of attraction and, in the theory of phase-locked loops, one will find methods to expand the region. Note also that, although integral compensation in $F(s)$ is not required in theory, we were unable to make the scheme work with simple proportional control ($F(s) = 1$).

VII. Improved Algorithm

An undesirable behavior was found to originate from the nonlinear dynamics associated with the parameter θ_1. Specifically, the parameter θ_1 sometimes converged to zero in simulations, with the parameter θ_2 becoming constant and the control input becoming zero. The problem is not specific to our implementation, but rather can be traced back to Fig. 1, which shows that $\theta_1 = 0$ is an equilibrium point of the closed-loop system. The approximate analysis also predicts this property.

Fortunately, the problem may be solved with an improved scheme, shown in Fig. 4. The signals w_1 and w_2 are similar to the previous signals, except that multiplication by θ_1 is eliminated in both loops. The equations for the control algorithm are the same as before, except for

$$u_q = -\sin(\alpha), \qquad u_c = \cos(\alpha),$$
$$w_1 = y \cdot y_c, \qquad y_c(s) = P_e(s)u_c(s).$$

As a result, the nonlinear approximation is now given by

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} = \tfrac{1}{2}P_M^2 \begin{bmatrix} \theta_1(t) - \theta_1^* \cos(\delta\alpha(t)) \\ \theta_1^* \sin(\delta\alpha(t)) \end{bmatrix} \tag{7}$$

and the equilibrium associated with $\theta_1 = 0$ is eliminated.

The responses of the improved algorithm for the same conditions as in Fig. 3 are faster for the same algorithm parameters (convergence within 0.6 instead of 1.2 s). For brevity, the corresponding figures are not shown. On the other hand, Fig. 5 is offered to show how the undesirable behavior is eliminated with the improved algorithm. The initial conditions are the same, except for $\theta_2(0) = 110$. The dashed line is for the original implementation, while the solid line is for the improved algorithm. The undesirable behavior of the original algorithm is visible, and is clearly avoided with the improved version.

One drawback of the improved algorithm is that three copies of the estimated plant $P_e(s)$ must be implemented. However, the third copy may be avoided by replacing y_e by $y_e(t) = \theta_1(t) \cdot y_c(t)$.

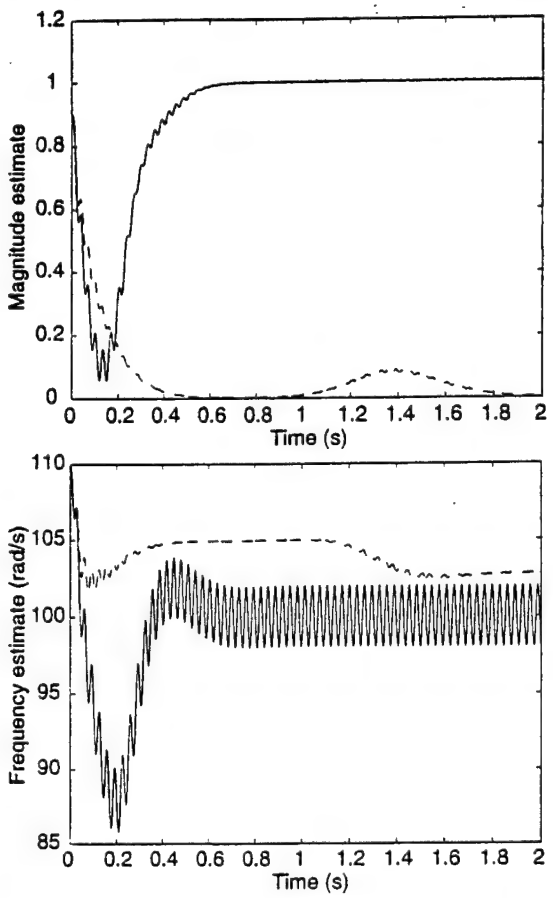

Fig. 5. Estimates θ_1 and θ_2 of the original and improved algorithms.

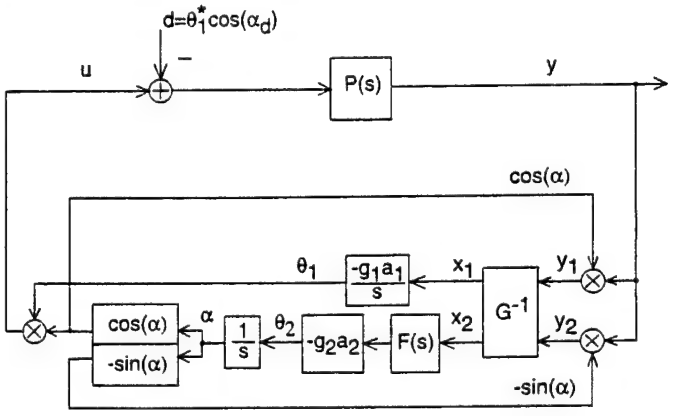

Fig. 6. Alternative algorithm.

While this definition of y_e is not identical to the previous one, it is equivalent under the assumption of slow variation of θ_1, and it yields the same approximate system. Expanding on that idea, one may further eliminate the other two copies of $P_e(s)$. Specifically, one may obtain y_c and y_q using the steady-state approximations, so that

$$\begin{pmatrix} y_c \\ y_q \end{pmatrix} = G^T \begin{pmatrix} \cos(\alpha) \\ -\sin(\alpha) \end{pmatrix}, \qquad G = \begin{pmatrix} P_R & -P_I \\ P_I & P_R \end{pmatrix}. \tag{8}$$

Again, these expressions are not identical to the previous expressions, but are equivalent as far as the approximate system is concerned.

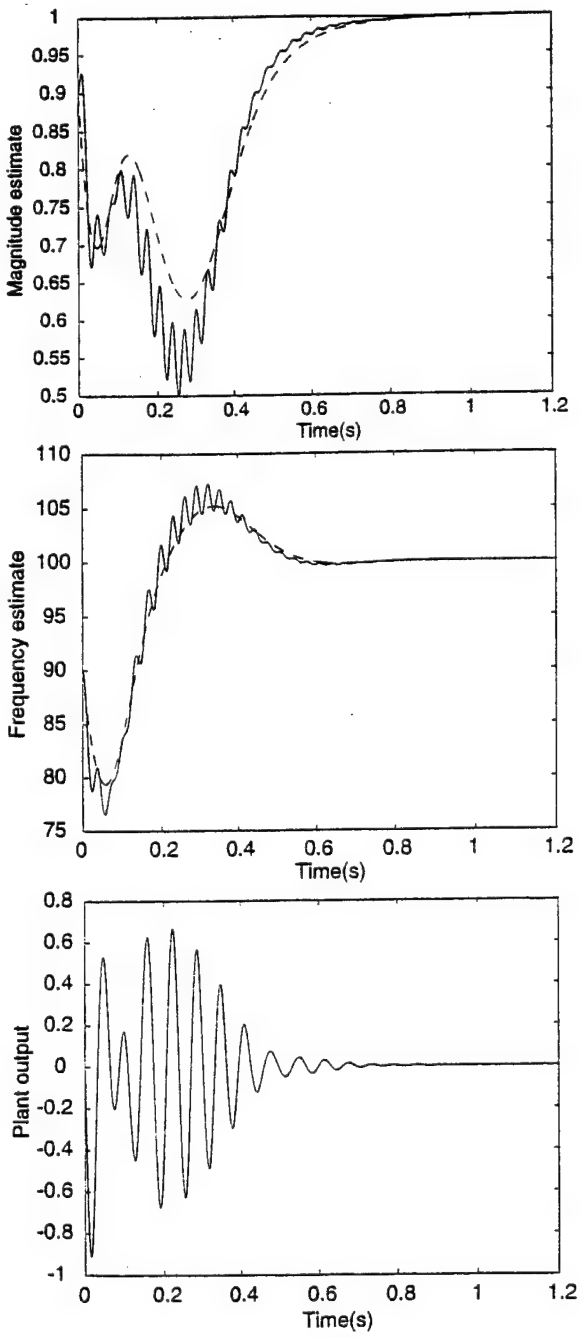

Fig. 7. Responses of the alternative algorithm.

VIII. Comparison with an Alternative Algorithm

The method of Chaplin and Smith presents some interesting similarities to a scheme proposed in [5]. For reference, Fig. 6 shows a schematic of the "direct" algorithm presented in [5]. The notation has been made similar to the notation of this paper for easy comparison. Further, similar choices were made for the design, so that the parameters could be set in an identical manner, except for the parameters a_1 and a_2, which are equal to two and to an estimate of $2/\theta_1^*$. Simulations for the alternative scheme are shown in Fig. 7. The responses are similar to those of the improved algorithm. In fact, the approximate system responses are identical. Of course, it should be noted that such a similarity was only obtained after considerable adjustments and redesigns made to the algorithm shown in Fig. 1, in view of the results obtained in [5].

Even so, there remains an interesting distinction between the two algorithms. Specifically, one finds in Fig. 3 that the frequency estimate oscillates around its nominal value in the steady state. As a result, a residual error is observed in the output y. Neither of these characteristics is found in the responses of the alternate scheme in Fig. 7.

The oscillation in the frequency estimate may be traced to the original diagram shown in Fig. 1, and is due to the signal at twice the frequency of the disturbance coming from the phase comparator. The oscillation could be reduced by lowering the bandwidth of the low-pass filter after the multiplier. However, it cannot be completely eliminated. For phase-locked loops used in communications systems, the bandwidth of the phase-locked loop may be three orders of magnitude smaller than the frequency of the parasitic signal. In control problems, the frequency separation is typically much smaller, especially if an objective is to have convergence of the system within a few cycles of the disturbance. While a filter could reduce the oscillation, it would also introduce a delay which would force a slower response of the control loop. In turn, the ability of the system to rapidly reject the disturbance and to track its variations would be reduced.

IX. Conclusions

In this paper, we discussed a feedback algorithm for the rejection of sinusoidal disturbances of unknown frequency. The algorithm was derived from a patent by Chaplin and Smith, specifying components that were not identified originally. Improvements were also proposed, which included a compensating filter with integral action and an implementation which eliminated an undesirable equilibrium state of the control system. An approximate analysis provided considerable help toward the selection of the parameters, and yielded stability as well as predictable performance. Overall, the results demonstrated that a working algorithm could be derived from the concepts of the patent, and described how it could be done. The scheme was also compared to another algorithm serving identical purposes, and while the structure and properties of the schemes were found to be similar, a significant difference was found in the steady-state characteristics of the algorithms. Although the patent concerns the specific problem of repetitive vibrations, the algorithms of the paper are applicable to a broad range of control problems where periodic disturbances of unknown frequency must be rejected.

References

[1] S. M. Kuo and D. R. Morgan, *Active Noise Control Systems*. New York: Wiley, 1996.

[2] K. S. Narendra and A. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[3] G. Feng and M. Palaniswamy, "A stable adaptive implementation of the internal model principle," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1220–1225, 1992.

[4] M. Bodson and S. Douglas, "Narrow-band disturbance rejection using adaptive feedback algorithms," in *Proc. SPIE Symp. Smart Structures and Mater.*, San Diego, CA, 1997, vol. 3039, pp. 45–56.

[5] ——, "Adaptive algorithms for the rejection of periodic disturbances with unknown frequency," *Automatica*, vol. 33, no. 12, pp. 2213–2221, 1997.

[6] G. B. B. Chaplin and R. A. Smith, "Method of and apparatus for cancelling vibrations from a source of repetitive vibrations," U.S. Patent 4 566 118, Jan. 21, 1986.

[7] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[8] J. R. Smith, *Modern Communications Circuits*. New York: McGraw-Hill, 1997.

Performance of an Adaptive Algorithm for Sinusoidal Disturbance Rejection in High Noise

Marc Bodson *
Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112, U.S.A.
(801) 581 8590 bodson@ee.utah.edu

Abstract

The paper proposes a discrete-time algorithm for the rejection of sinusoidal disturbances of unknown frequency. The algorithm is adapted from an existing continuous-time algorithm and the overall nonlinear system is analyzed using a linear approximation. The paper shows that the noise rejection properties may be predicted using an in-phase/quadrature decomposition of the noise similar to the one encountered in the theory of phase-locked loops. Estimates are obtained for the standard deviations of the plant output and of the adaptive parameters and simulations validate the predictions of the analysis, despite the nonlinear nature of the adaptive system and the high level of noise applied.

1 Introduction

The rejection of periodic disturbances is a common problem in control applications. If the frequency of the disturbance is known, several techniques are available, including internal model control, adaptive methods, and repetitive control techniques. These methods are closely connected. In particular, it was shown in [1] that a standard adaptive feedforward control algorithm was equivalent to an internal model control law. Examples of applications include helicopters [2] and high-density magnetic disk drives [3].

When the frequency of the disturbance is not known, the problem is considerably more complicated. Sometimes, the disturbance (or a signal related to it) can be measured, and the problem can be solved using an adaptive feedforward control algorithm. However, in many applications, it is inconvenient, if not impossible, to place a sensor on the source of the disturbance or along its path to the plant. For such problems, an adaptive implementation of the internal model principle may be considered [4],[5], [6]. Such algorithms are highly nonlinear and, aside from Lyapunov-type stability results, little is known about their convergence properties or their sensitivity to measurement noise.

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

In [7], two adaptive algorithms were proposed for the cancellation of sinusoidal disturbances with unknown frequency. The first algorithm was called *indirect*, and was obtained by combining a frequency estimation algorithm together with an adaptive algorithm for the cancellation of disturbances of known frequency. A comparable approach was followed in [8], using a different frequency estimation algorithm and a repetitive control method. The second algorithm of [7] was called *direct*, and estimated the frequency, phase, and magnitude of the disturbance in an integrated fashion. The scheme was obtained by combining a phase-locked loop concept with a sinusoidal disturbance cancellation scheme. The algorithm was extended to handle multiple sinusoidal components and implemented successfully in an active noise control testbed [9].

As other algorithms for the rejection of periodic disturbances of unknown frequency, the direct algorithm of [7] is highly nonlinear. However, it may be approximated by a linear system using techniques similar to those used for phase-locked loops [10]. As a result, estimates of convergence rates can be obtained that are useful for design. The contribution of this paper is to show that the continuous-time algorithm of [7] may be adapted to discrete-time, and that its noise properties may be predicted using an in-phase/quadrature decomposition of the measurement noise signal. Simulations show that the results of the analysis are accurate, even in the presence of high noise.

2 A Discrete-Time Algorithm

2.1 Problem Statement

The plant is assumed to be described, in the z-transform domain, by

$$\begin{aligned} y(z) &= P(z)(u(z) - d(z)) \\ \bar{y}(z) &= y(z) + n(z) \end{aligned} \tag{1}$$

where u is the control input, d is the disturbance, n is the measurement noise, y is the plant output, and $\bar{y}$ is the measured plant output. $P(z)$ is the transfer function of the plant, which is assumed to be stable. In the time-domain, the disturbance is assumed to be given by

$$d(k) = d_1 \cos(\alpha_d(k)), \tag{2}$$

with

$$\alpha_d(k) = \omega_1 k + \delta_1. \tag{3}$$

The disturbance is a sinusoidal function with magnitude d_1, frequency ω_1, and initial phase δ_1. These parameters are fixed, but unknown. Alternatively, (3) can be written in a recursive form as

$$\alpha_d(k+1) = \alpha_d(k) + \omega_1, \quad \alpha_d(0) = \delta_1. \tag{4}$$

Although the disturbance does not need to act at the input of the plant, it is assumed that an equivalent input disturbance may be so defined. For its cancellation, the input is then chosen to be of the form

$$u(k) = \theta_1(k) \cos(\alpha(k)), \tag{5}$$

with

$$\alpha(k+1) = \alpha(k) + \theta_2(k), \quad \alpha(0) = 0. \tag{6}$$

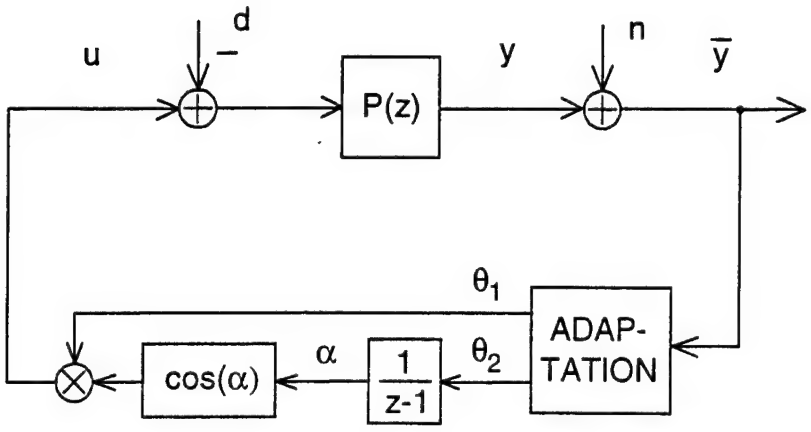

Figure 1: Plant and Adaptive System

The variables $\theta_1(k)$ and $\theta_2(k)$ are parameters with nominal values $\theta_1^* = d_1$ and $\theta_2^* = \omega_1$. The phase of the disturbance is estimated through the variable $\alpha(k)$, and it is determined by the time history of $\theta_2(k)$ rather than through separate adaptation: even though the problem is stated in terms of three unknown parameters, the solution is based on the adaptation of only two parameters $\theta_1(k)$ and $\theta_2(k)$. Specifically, for the exponentially stable adaptive scheme developed in this paper, the phase δ_1 is determined through

$$\delta_1 = \sum_{k=0}^{\infty} (\theta_2(k) - \omega_1) \tag{7}$$

The structure of the overall system is shown in Fig. 1.

2.2 Basic Results

In a first step, we assume that the measurement noise $n = 0$, so that $y = \bar{y}$. The algorithm is based on the following fact.

Fact:

Assume that:

- θ_1 and θ_2 vary sufficiently slowly that the response of the plant to the signal u can be approximated by the steady-state output of the plant for a sinusoidal input with frequency θ_2.
- the instantaneous frequency θ_2 is close to ω_1, so that $P(e^{j\theta_2})$ can be replaced by $P(e^{j\omega_1})$.
- the phase error $\alpha - \alpha_d$ is small.

Then: considering low-frequency components only, the two signals

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} y(k)\cos(\alpha(k)) \\ -y(k)\sin(\alpha(k)) \end{bmatrix} \tag{8}$$

are approximately given by

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = G \begin{bmatrix} \theta_1(k) - d_1 \\ d_1(\alpha(k) - \alpha_d(k)) \end{bmatrix}, \tag{9}$$

where

$$G = \frac{1}{2} \begin{bmatrix} Re[P(e^{j\omega_1})] & -Im[P(e^{j\omega_1})] \\ Im[P(e^{j\omega_1})] & Re[P(e^{j\omega_1})] \end{bmatrix}.$$

Proof: Let

$$P_R = Re[P(e^{j\omega_1})], \quad P_I = Im[P(e^{j\omega_1})]. \tag{10}$$

Under the assumptions, the output of the plant is given by

$$\begin{aligned} y(k) &= P_R\theta_1(k)\cos(\alpha(k)) - P_I\theta_1(k)\sin(\alpha(k)) \\ &- P_R d_1 \cos(\alpha_d(k)) + P_I d_1 \sin(\alpha_d(k)). \end{aligned} \tag{11}$$

Keeping only the low-frequency components of the signals y_1 and y_2, we have

$$\begin{aligned} y_1(k) &= \frac{1}{2}P_R\theta_1(k) - \frac{1}{2}P_R d_1 \cos(\alpha(k) - \alpha_d(k)) \\ &- \frac{1}{2}P_I d_1 \sin(\alpha(k) - \alpha_d(k)), \\ y_2(k) &= \frac{1}{2}P_I\theta_1(k) + \frac{1}{2}P_R d_1 \sin(\alpha(k) - \alpha_d(k)) \\ &- \frac{1}{2}P_I d_1 \cos(\alpha(k) - \alpha_d(k)). \end{aligned} \tag{12}$$

Assuming that the phase error $\alpha - \alpha_d$ is small, the result is obtained.
Comments: Equation (9) can be viewed as an alternative description of the plant. Note that

$$\alpha(k) - \alpha_d(k) = \alpha(k-1) - \alpha_d(k-1) + \theta_2(k-1) - \omega_1 \tag{13}$$

so that, with the change of variables, the plant is approximately a linear time-invariant plant with two inputs $\theta_1(k)$ and $\theta_2(k)$ and two outputs $y_1(k)$ and $y_2(k)$. The transfer function matrix consists of a gain matrix G and a discrete-time integrator in the second channel. The parameters d_1 and ω_1 act as constant disturbances applied to the inputs. The parameter d_1 also appears as a gain in the second channel. In contrast to (9), equation (12) constitutes a *nonlinear* approximation of the plant. This nonlinear approximation is useful to understand the transient properties of the algorithm, but is not used in this paper.

The elimination of the high-frequency components in the signals $y_1(k)$ and $y_2(k)$ can be achieved by low-pass filtering. In the algorithm discussed in this paper, the signals are applied to a compensator which is itself low-pass. Thus, no filter was used for the simulations. However, it may be added if needed.

2.3 Compensator Design

Define two variables w_1, w_2 through

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = G^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \tag{14}$$

so that the dynamics of the system become decoupled. Specifically, in the z-domain

$$\begin{aligned} w_1 &= \theta_1 - \theta_1{}^* \\ w_2 &= \frac{d_1}{z-1}\,[\theta_2 - \theta_2{}^*]. \end{aligned} \tag{15}$$

The control law is chosen to be the cascade of the transformation (14) and compensators

$$\theta_1 = \frac{C_1(z)}{z-1}[w_1], \qquad \theta_2 = \frac{C_2(z)}{z-1}[w_2]. \tag{16}$$

The poles at $z = 1$ are included in the compensators to guarantee the rejection of the constant disturbances composed of d_1 and ω_1 (the compensators are selected as conventional controllers with integral compensation). The compensator transfer functions $C_1(z)$ and $C_2(z)$ are designed to guarantee the stability of the closed-loop systems and satisfactory transient responses. For example, one can choose

$$C_1(z) = -g_1, \qquad C_2(z) = -g_2\frac{z-z_a}{z-z_b}, \tag{17}$$

where g_1, g_2, z_a, and z_b are design parameters. The overall adaptive algorithm is then given by (5), (6), (8), (14), (16), and (17).

Two simple tuning methods are proposed for the parameters of the control law. The first is the equivalent of the continuous-time control law of [7]. Let z_d be some desirable location to place the closed-loop poles. For the first channel, the objective leads to a parameter $g_1 = 1 - z_d$. For the second channel, the parameters are

$$g_2 = \frac{3(1-z_d)^2}{d_1}, \qquad z_a = \frac{z_d+2}{3}, \qquad z_b = 3z_d - 2. \tag{18}$$

The second method is an alternative proposed specifically for the discrete-time version of the algorithm. It consists in placing the pole of the compensator for the second channel at the origin and neglecting the effect of the pole. The remaining closed-loop poles are both placed at z_d, giving the following parameters

$$g_2 = \frac{2(1-z_d)}{d_1}, \qquad z_a = \frac{z_d+1}{2}, \qquad z_b = 0. \tag{19}$$

The response of the algorithm with the second tuning method was found to be faster and with a wider convergence range, but also with a higher sensitivity to noise.

Estimates of the magnitude of the disturbance and of its frequency are required. These estimates are used as initial conditions for the variables θ_1 and θ_2, and also for the design of the compensators. The estimate of the magnitude of the disturbance d_1 is used to adjust the gain g_2 of the compensator $C_2(z)$ and the estimate of the frequency of the disturbance is used to adjust the gain matrix G^{-1}. The variables g_2 and G^{-1} may be kept constant during operation, or they may be updated as functions of the variables θ_1 and θ_2. In experiments with this algorithm [9], the frequency response of the plant was estimated automatically in an initial tuning phase, and the matrix G was updated by linear interpolation of a table look-up as a function of the estimated frequency. In the simulations of this paper, fixed values were used. Generally, the frequency of the disturbance may be estimated using an estimation algorithm (such as the one discussed in [7], or those of [12]). The magnitude may be estimated using the measured RMS output, or a fast Fourier transform. However, the region of convergence of the adaptive scheme is generally large enough that estimates based on prior information are typically sufficient.

3 Noise Analysis

3.1 Linear Approximation

The previous derivation considered the noiseless case, which resulted in nominal values for the parameters $\theta_1^* = d_1$, $\theta_2^* = \omega_1$, and nominal functions $\alpha^*(k) = \alpha_d(k)$, $u^*(k) = d_1 \cos(\alpha_d(k))$, $y^*(k) = 0$. Now, the effect of a measurement noise $n(k)$ added to the output $y(k)$ is evaluated. The idea of the derivation is to assume that the noise is sufficiently small that second-order effects can be neglected and to decompose the noise into in-phase and quadrature components.

Define $\theta_1 = \theta_1^* + \delta\theta_1$, $\theta_2 = \theta_2^* + \delta\theta_2$, $\alpha = \alpha^* + \delta\alpha$, $u = u^* + \delta u$, $y = y^* + \delta y$, $\bar{y} = y^* + \delta y + n$, where y is the actual output of the plant, and $\bar{y}$ is the measured output (the one used by the algorithm). Since $y^* = 0$, $y = \delta y$ and $\bar{y} = \delta\bar{y}$, where $\delta\bar{y} = \delta y + n$. Define components of the measured output

$$\begin{bmatrix} \bar{y}_1(k) \\ \bar{y}_2(k) \end{bmatrix} = \begin{bmatrix} \bar{y}(k)\cos(\alpha(k)) \\ -\bar{y}(k)\sin(\alpha(k)) \end{bmatrix}. \tag{20}$$

Again, these two variables are the ones that are used by the algorithm, instead of y_1, y_2, which are not available but remain as defined previously for the purpose of analysis. For the noise, we introduce the *in-phase* and *quadrature* components

$$\begin{bmatrix} n_1(k) \\ n_2(k) \end{bmatrix} = \begin{bmatrix} n(k)\cos(\alpha_d(k)) \\ -n(k)\sin(\alpha_d(k)) \end{bmatrix}. \tag{21}$$

From (20),

$$\begin{aligned} \bar{y}_1(k) &= (y(k) + n(k))\cos(\alpha_d(k) + \delta\alpha(k)), \\ \bar{y}_2(k) &= -(y(k) + n(k))\sin(\alpha_d(k) + \delta\alpha(k)). \end{aligned} \tag{22}$$

Assuming that the noise is small and neglecting second-order effects, it follows that

$$\begin{aligned} \bar{y}_1(k) &= y(k)\cos(\alpha_d(k)) + n_1(k), \\ \bar{y}_2(k) &= -y(k)\sin(\alpha_d(k)) + n_2(k). \end{aligned} \tag{23}$$

The control input is given by

$$u(k) = (d_1 + \delta\theta_1(k))\cos(\alpha_d(k) + \delta\alpha(k)), \tag{24}$$

so that, approximately,

$$\delta u(k) = \cos(\alpha_d(k))\delta\theta_1(k) - d_1\sin(\alpha_d(k))\delta\alpha(k). \tag{25}$$

Assuming slow variations of $\delta\theta_1$ and $\delta\alpha$, the output of the plant corresponding to this input is given by

$$\begin{aligned} y(k) &= P_R\cos(\alpha_d(k))\delta\theta_1(k) - P_R d_1\sin(\alpha_d(k))\delta\alpha(k) \\ &\quad -P_I\sin(\alpha_d(k))\delta\theta_1(k) - P_I d_1\cos(\alpha_d(k))\delta\alpha(k). \end{aligned} \tag{26}$$

Multiplication of this signal by $\cos(\alpha_d)$ and $\sin(\alpha_d)$, and application to (23), yields signals whose first-order/low-frequency components are given by

$$\begin{bmatrix} \bar{y}_1(k) \\ \bar{y}_2(k) \end{bmatrix} = G \begin{bmatrix} \delta\theta_1(k) \\ d_1\delta\alpha(k) \end{bmatrix} + \begin{bmatrix} n_1(k) \\ n_2(k) \end{bmatrix}. \tag{27}$$

The result shows that, to first-order, the effect of the measurement noise is equivalent to the addition of two noise variables in the transformed system. Again, analysis can again be performed in a linear time-invariant framework.

3.2 Computation of Variances

A transfer function matrix relates the noise sources to the parameter deviations, with

$$\begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \delta\alpha \end{bmatrix} = \begin{bmatrix} \frac{C_1(z)}{z-1-C_1(z)} & 0 \\ 0 & \frac{(z-1)C_2(z)}{(z-1)^2-d_1C_2(z)} \\ 0 & \frac{C_2(z)}{(z-1)^2-d_1C_2(z)} \end{bmatrix} G^{-1} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}. \tag{28}$$

To apply the results and estimate the performance of the algorithm in the presence of noise, the power spectra of the noise components n_1 and n_2 must be known. If n is a white noise with variance σ^2, it is common to assume, in the analysis of phase-locked loops, that n_1 and n_2 are uncorrelated white noises with variances equal to $\frac{1}{2}\sigma^2$ [10]. We will make the same assumption. The power spectra of $\delta\theta_1$, $\delta\theta_2$, and $\delta\alpha$ can then be deduced using the transfer function (28), and the variances of these variables can be computed by integration of the power spectra.

Alternatively, a Lyapunov equation may be solved, using the following state-space description

$$x(k+1) = Ax(k) + BG^{-1}v(k), \tag{29}$$

where

$$\begin{aligned} A &= \begin{bmatrix} 1-g_1 & 0 & 0 & 0 \\ 0 & 1+z_b & 1 & -g_2d_1 \\ 0 & -z_b & 0 & g_2z_ad_1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -g_1 & 0 \\ 0 & -g_2 \\ 0 & g_2z_a \\ 0 & 0 \end{bmatrix}, \\ x(k) &= \begin{bmatrix} \delta\theta_1(k) \\ \delta\theta_2(k) \\ \delta\theta_3(k) \\ \delta\alpha(k) \end{bmatrix}, \quad v(k) = \begin{bmatrix} n_1(k) \\ n_2(k) \end{bmatrix}. \end{aligned} \tag{30}$$

Note that these equations, as well as previous equations, assume that the G matrix used by the compensator is the true matrix corresponding ot the plant. However, the analysis can be extended to account for a difference in the true and estimated matrices. Defining $V = E(vv^T) = \frac{1}{2}\sigma^2 I$, and assuming that n_1 and n_2 are white noise sources, the value of the steady-state covariance matrix $P = E(xx^T)$ is given by the solution of the discrete Lyapunov equation ([11], p. 471)

$$AXA^T + BG^{-1}V(G^{-1})^TB^T = X. \tag{31}$$

which is easily obtained using modern mathematical software.

Regarding the output signals y and $\bar{y}$, (26) shows that y is a sinusoidal function with the same frequency as the disturbance, but with a peak and phase that are random processes. The variance of the magnitude of the sinusoid is equal to $(P_R{}^2+P_I{}^2)(E(\delta\theta_1{}^2)+d_1{}^2E(\delta\alpha^2))$. Although the variance of the plant output is a function of time, its average over the period of the disturbance may be defined. This average variance may be estimated experimentally or in simulations, and is a significant performance measure for the algorithm. Using the previous results, one finds that the average variances of the true and measured plant outputs are given by

$$\begin{aligned} E_{avg}(y^2) &= \frac{(P_R{}^2+P_I{}^2)}{2}(E(\delta\theta_1{}^2)+d_1{}^2E(\delta\alpha^2)) \\ E_{avg}(\bar{y}^2) &= E_{avg}(y^2)+\sigma^2. \end{aligned} \tag{32}$$

The subscript avg refers to the averaging of the variances over the period of $\alpha_d(k)$.

4 Simulation Results

We present simulation results obtained for a plant which is a pure delay $P(z)=z^{-10}$. The disturbance has magnitude $d_1=1$, a period equal to 100 time samples,. The initial phase was set at $\delta_1=0$, but could be set at arbitrary values. The estimates used by the algorithm are 0.8 for the magnitude and 120 steps for the period. The desired closed-loop pole was selected to be $z_d=0.99$, leading to $g_1=0.01$, $g_2=0.025$, $z_a=0.995$, and $z_b=0$ (for the second tuning method). These parameters were left fixed throughout the simulations.

Fig. 2 shows the measured output of the plant ($\bar{y}$). The plot on the left corresponds to a simulation for a low noise condition: $\sigma=0.01$, with the uncompensated plant output being a sinusoid with magnitude equal to 1. The plot on the right corresponds to a high noise condition, with $\sigma=0.5$ (or a standard deviation equal to 50% of the magnitude of the uncompensated plant output, instead of 1%). Figs. 3 and 4 show the responses of the magnitude and frequency estimates (θ_1 and θ_2), showing convergence towards the expected values (the nominal magnitude is 1 and the nominal frequency is $2\pi/100=0.0628$). In the high noise plots, the convergence of the parameters towards their nominal values is barely visible in the fluctuations induced by the noise. However, the output is significantly smaller than what it would be under uncompensated conditions, as shown in Fig. 5.

The predicted variances were computed by solving the discrete Lyapunov equation. The results were compared to sample variances obtained from the simulations by averaging deviations between $k=1000$ and $k=11000$ (in other words, a longer time period was used than for the plots, and the initial transient was left out of the computations). The results are summarized in Table 1. The standard deviations predicted by the solution of the Lyapunov equation are given under the heading of "analysis", while the results obtained through averaging of the simulations results are shown as "simulation." The numbers show a good match between the predictions of the analysis and the values observed in simulations. One also finds that the variation of the measured output is primarily made of the noise itself, rather than a variation induced by the fluctuations of the adaptive parameters.

Although the analysis of the paper and the simulations emphasized the steady-state performance of the algorithm, significant changes in the disturbance parameters, as well as abrupt changes, may be accommodated by the algorithm. Fig. 6 shows the results of a simulation with low noise over 3000 samples, with the magnitude and the frequency of the disturbance increased by 50% at $k=1000$

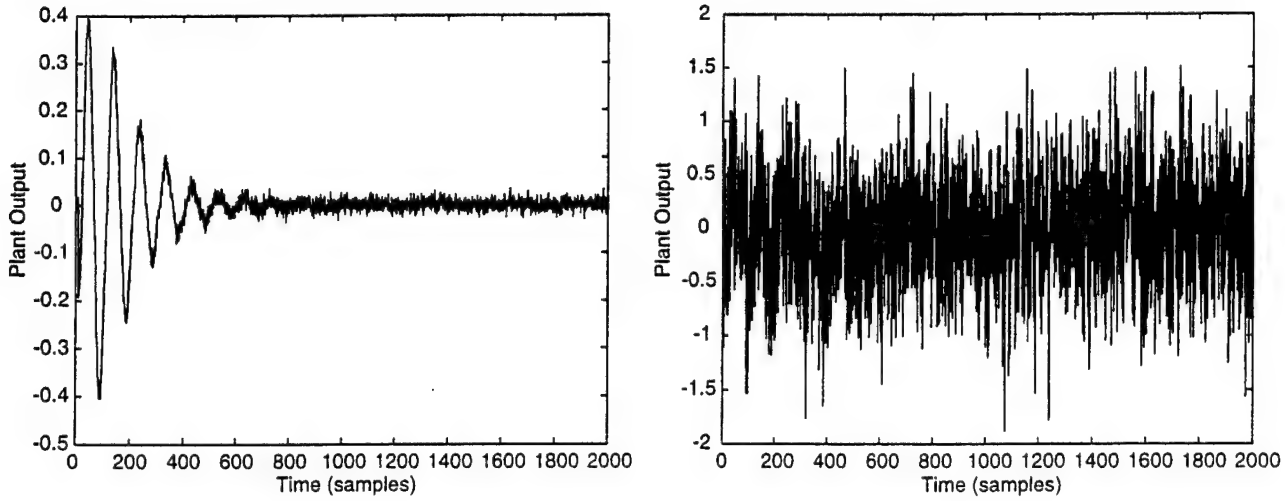

Figure 2: Measured Plant Output - Low Noise (Left), High Noise (Right)

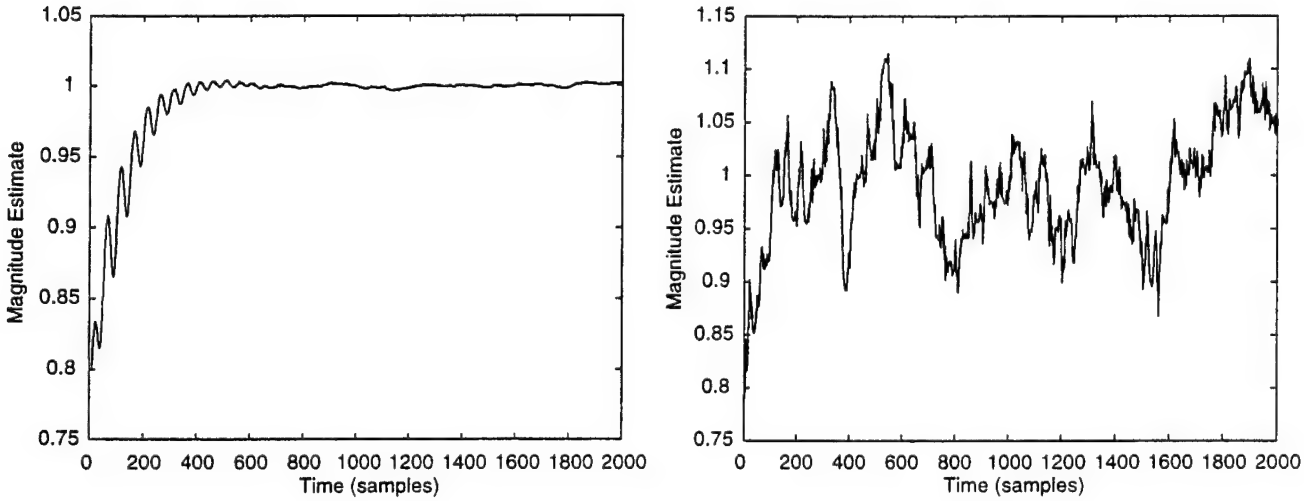

Figure 3: Magnitude Estimate - Low Noise (Left), High Noise (Right)

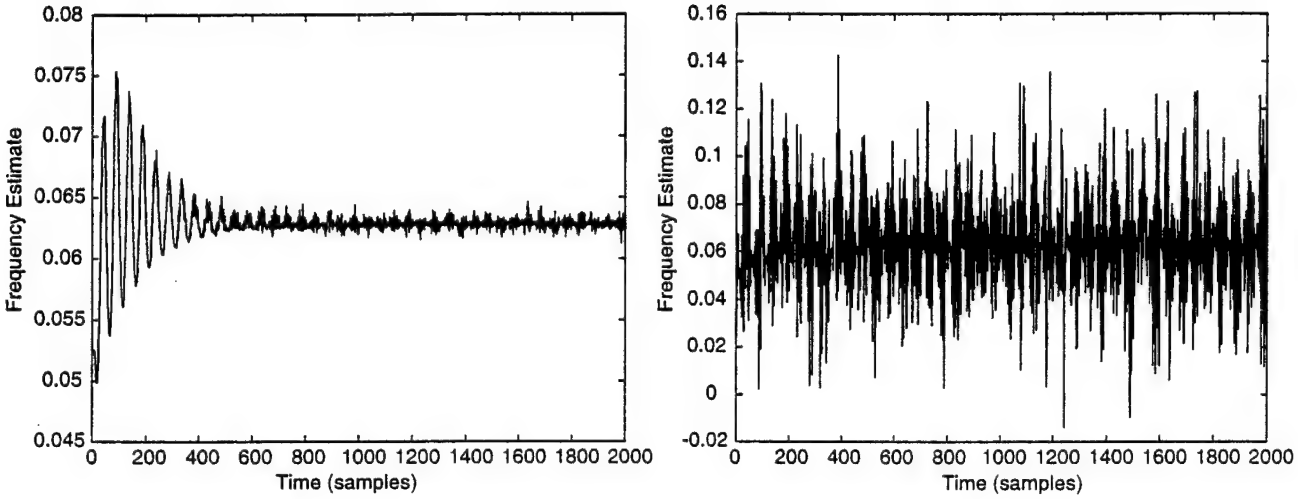

Figure 4: Frequency Estimate - Low Noise (Left), High Noise (Right)

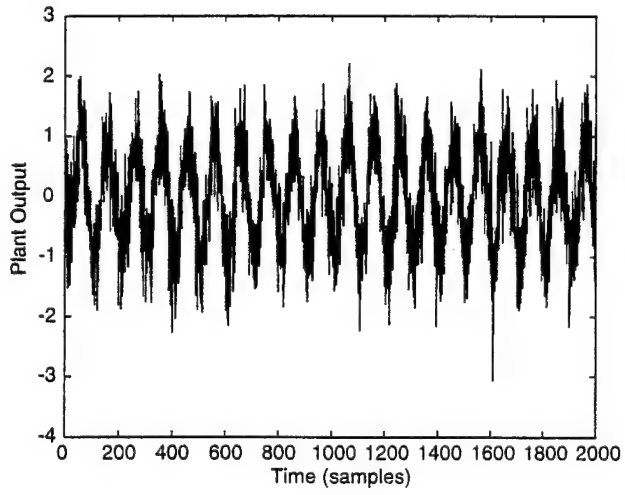

Figure 5: Measured Plant Output without Compensation and with High Noise

	y	$\bar{y}$	$\delta\theta_1$	$\delta\theta_2$
$\sigma = 0.01$				
Analysis	0.0014	0.0101	0.0010	3.5610^{-4}
Simulation	0.0016	0.0103	0.0011	3.6510^{-4}
$\sigma = 0.5$				
Analysis	0.0718	0.5051	0.0501	0.0178
Simulation	0.0881	0.5110	0.0613	0.0180

Table 1: Standard Deviations of Plant Output and Parameters – Analysis *vs.* Simulation

and $k = 2000$, respectively. At $k = 1000$, the phase also jumps by 180° (the maximum possible). The figure shows the response of the magnitude estimate (left) and of the frequency estimate (right). After some transient responses, both estimates converge to their desired values.

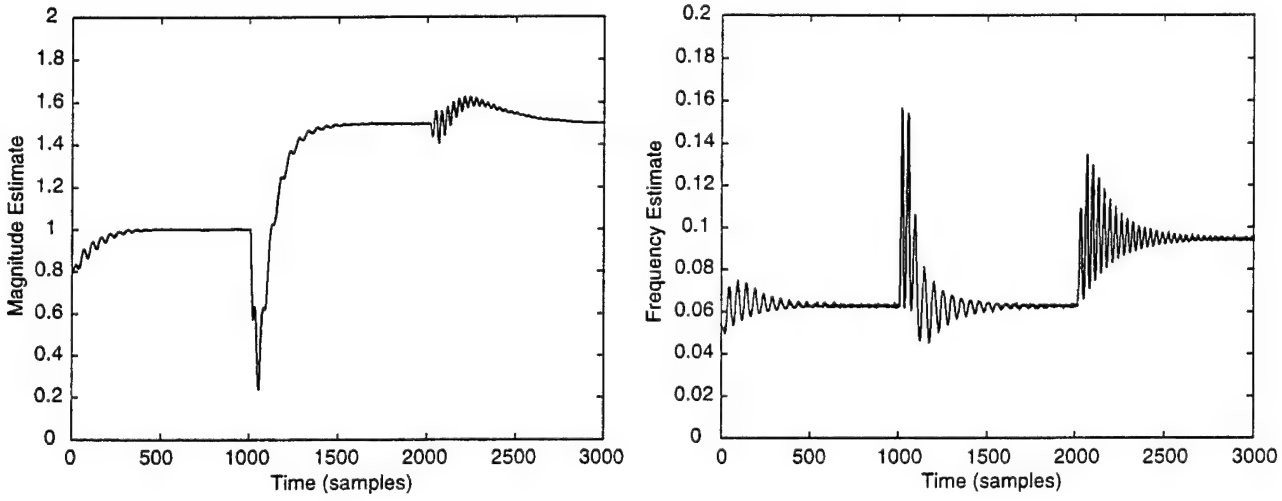

Figure 6: Adaptive Parameter Responses to Sudden Changes in the Disturbance Parameters

5 Conclusions

In this paper, we discussed a method for the rejection of sinusoidal disturbances with unknown frequency. The discrete-time algorithm was obtained from an existing continuous-time algorithm, with some non-trivial adjustments. Further extensions to handle multiple harmonics are possible. The design of the control law could be conveniently carried out using linear techniques. In addition, a separate analysis predicted the loss of performance incurred in the presence of measurement noise, and the amount of fluctuation induced in the adaptive parameters. Through simulations, it was found that the algorithm was not only effective at rejecting sinusoidal disturbances in the presence of high

noise, but also that its performance could be accurately predicted. The derivations relied on a phase-locked loop technique that decomposes the noise signal into in-phase and quadrature components. Such an approach has not been widely used in control, but was found helpful for this problem. The results that were obtained do not have counterparts for other algorithms proposed in the literature for the rejection of periodic disturbances of unknown frequency.

References

[1] M. Bodson, A. Sacks & P. Khosla, "Harmonic Generation in Adaptive Feedforward Cancellation Schemes," *IEEE Trans. on Automatic Control*, vol. 39, no. 9, 1994, pp. 1939-1944.

[2] S.R. Hall & N.M. Wereley, "Performance of Higher Harmonic Control Algorithms for Helicopter Vibration Reduction," *J. Guidance, Control and Dynamics,* vol. 16, no. 4, 1993, pp. 793-797.

[3] A. Sacks, M. Bodson, & P. Khosla, "Experimental Results of Adaptive Periodic Disturbance Cancellation in a High Performance Magnetic Disk Drive," *ASME Journal on Dynamics and Control,* vol. 118, 1996, pp. 416-424.

[4] K.S. Narendra and A. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[5] G. Feng and M. Palaniswamy, "A Stable Adaptive Implementation of the Internal Model Principle," *IEEE Trans. on Automatic Control*, vol. 37, no. 8, 1992, pp. 1220-1225.

[6] B. Francis & M. Vidyasagar, "Linear Multivariable Regulation with Adaptation: Tracking Signals Generated by Models with Unknown Parameters", *Proc. of the IEEE Conference on Decision and Control,* 1978.

[7] M. Bodson & S. Douglas, "Adaptive Algorithms for the Rejection of Periodic Disturbances with Unknown Frequency," *Automatica,* vol. 33, no. 12, 1997, pp. 2213-2221.

[8] T.J. Manayathara, T.-C. Tsao, J. Bentsman, & D. Ross, "Rejection of Unknown Periodic Load Disturbances in Continuous Steel Casting Process Using Learning Repetitive Control Approach," *IEEE Trans. on Control Systems Technology,* vol. 4, no. 3, 1996, pp. 259-265.

[9] M. Bodson, J. Jensen, & S. Douglas, "Active Noise Control for Periodic Disturbances," *Proc. of the American Control Conference,* Philadelphia, PA, 1998, pp. 2616-2620.

[10] D.H. Wolaver, *Phase-Locked Loop Circuit Design,* Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[11] H. Kwakernaak & R. Sivan, *Linear Optimal Control Systems*, Wiley, New York, 1972

[12] P. Stoica & R. Moses, *Introduction to Spectral Analysis*, Prentice-Hall, Upper Saddle River, NJ, 1997.

Proceedings of the 38th
Conference on Decision & Control
Phoenix, Arizona USA • December 1999

Multi-Channel Active Noise Control for Periodic Disturbances

Biqing Wu and Marc Bodson *
Department of Electrical Engineering, University of Utah
Salt Lake City, UT 84112, U.S.A.

Abstract: The paper presents a multi-channel active noise control algorithm that is designed to reject periodic noise signals of unknown frequency. The algorithm is the extension of a previously-proposed disturbance rejection algorithm in which the frequency of the disturbance is estimated through an adaptive notch filter and the estimate is used in a feedforward disturbance rejection scheme. Improvements over the earlier approach include a better frequency estimation scheme and an extension to multi-channel systems. Experimental results on an active noise control testbed demonstrate the validity of the analytical results and the success of the method in a practical environment.

1. Introduction

The system for multi-channel active noise control (ANC) is shown in Fig. 1. Microphones sense the effect of the noise source and monitor how well the control system is performing. The microphone signals are then processed by a digital signal processing system, and an anti-noise field is generated through speakers. The dynamics of the sound propagation from the speakers to the microphones constitutes the plant. The objective of the ANC system is to eliminate or significantly reduce the noise level at the microphone locations through destructive interference[1].

Multi-channel (or multiple-input/multiple-output) control is needed for the rejection of the noise at multiple locations in a 3-dimensional space. Compared with single-channel ANC system, the complexity of multi-channel ANC's is significantly higher. Also, noise generated by engines, compressors, fans, and propellers is periodic. The frequency of the noise is not fixed *a priori* and is often time-varying. Therefore, the algorithm discussed in this paper is tailored to periodic disturbances of unknown frequency. The problem was considered in [2], where a direct approach and an indirect approach were proposed. The direct approach was successfully tested in a single-channel system, with the results reported in [3]. The indirect approach was found to suffer from poor convergence properties, but a fix was later found and simulations for a single-channel system showed promising results [4]. This paper extends the indirect adaptive algorithm of [2] and [4] to a multi-channel system and reports experimental results. The results show that, despite high initial uncertainty, the multi-channel control algorithm is able to reduce significantly the noise level at the microphone locations. In addition, the algorithm converges rapidly.

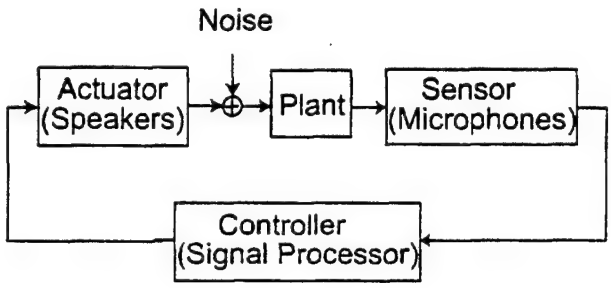

Figure 1: Active Noise Control System

2. Multi-Channel Adaptive Algorithm

Problem Statement

Assume that the effect of the noise is additive and that the channels from the speakers to the microphones can be described by stable, linear time-invariant systems with transfer functions $P_{ij}(s)$, where $i = 1, 2$ stands for microphones #1,2 and $j = 1, 2$ stands for speakers #1,2. For simplicity, a 2×2 system is considered. However, the techniques may be extended to systems of higher dimensions. Fig. 2 shows the structure of the two-input two-output system, where $u_i(s)$, $y_i(s)$, and $d_i(s)$, are the Laplace transforms of the speaker signals, the microphone signals, and the disturbance signals, respectively. Define the vector of plant outputs $Y(s)$, so that

$$Y(s) = \begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} u_1(s) - d_1(s) \\ u_2(s) - d_2(s) \end{bmatrix} .$$

The objective of the control system is to generate $u_1(t)$ and $u_2(t)$ such that $Y(t) \to 0$, as $t \to \infty$, *i.e.*, such that the effects of the disturbance signals at the two microphone locations are eliminated.

The disturbance signals $d_{1,2}(s)$ are assumed to be sinusoidal signals with unknown frequency. The frequency is assumed to be the same for both signals, although the amplitudes and phases are assumed to be independent. The disturbance signals are represented by

$$d_1(t) = \theta^*_{c1} \cos(\omega^* t) - \theta^*_{s1} \sin(\omega^* t) ,$$

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-96-1-0085. The content of the paper does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

0-7803-5250-5/99/$10.00 © 1999 IEEE

$$d_2(t) = \theta^*_{c2}\cos(\omega^* t) - \theta^*_{s2}\sin(\omega^* t)\ ,$$

where ω^* is the unknown frequency of the disturbance. In addition to the frequency, the unknown characteristics of the disturbance signals are parameterized as the amplitudes of the cos and sin components. This parameterization enables the use of a standard adaptive cancellation algorithm for the disturbance rejection. Note that although a single sinusoidal component of the disturbance is assumed, additional harmonic components may easily be added to the formulation of the problem as well as to the algorithm.

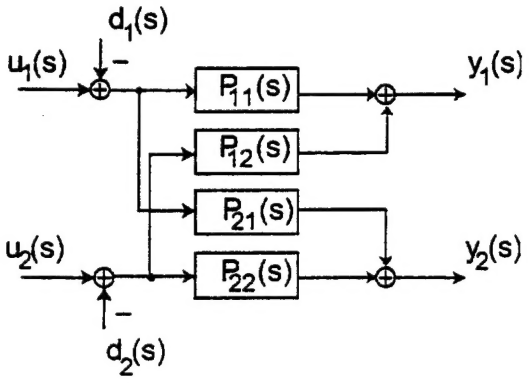

Figure 2: Multi-Channel Noise Control Problem

Frequency Estimation

The indirect approach to active noise control [2] consists of estimating the frequency of the sinusoidal disturbance and using the estimate in an adaptive disturbance rejection scheme for known frequency. In simulations, it was found that the adaptive notch filter developed by Regalia [5] performed well. In [2], a continuous-time version of Regalia's algorithm was studied using averaging analysis, and good results were obtained for the estimation of the frequency of a signal. However, poor convergence properties were observed when combining the estimation algorithm with a disturbance rejection algorithm. The problem was traced to the fact that the disturbance rejection algorithm eliminated the signal upon which frequency estimation was based, resulting in a slow convergence of the overall scheme.

In [4], a fix was found to resolve the problem. The fix simply consisted of subtracting the effect of the control signals on the microphones. Fig. 3 shows the implementation of the concept on the 2-channel system. $\hat{P}_{11}(s)$ and $\hat{P}_{12}(s)$ are estimates of $P_{11}(s)$ and $P_{12}(s)$, respectively. ANF refers to the adaptive notch filter and ω to the estimate of the frequency. Only one of the microphone signals is used for the frequency estimate in this implementation. The effect of the control signals $u_1(t)$ and $u_2(t)$ on the output of the plant is computed as an output signal y_{1u} which is subtracted from the microphone signal before the resulting signal is applied the adaptive notch filter. This modification results in better frequency estimation, which is critical in the practical implementation and improves the stability of the overall system. Surprisingly, it was found that the estimates of the plant transfer function did not have to be precise in order for the approach to work well.

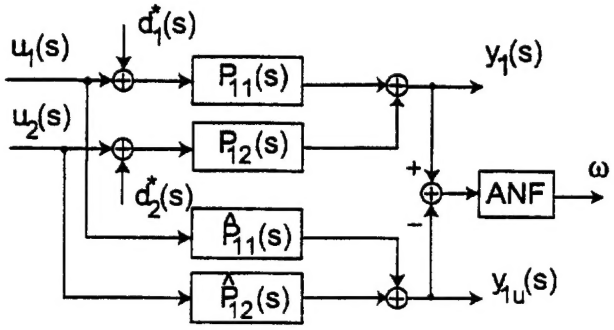

Figure 3: Frequency Estimation Scheme

Disturbance Cancellation

The second component of the indirect adaptive algorithm is a standard adaptive cancellation algorithm ([2], [6]), extended for the multi-input, multi-output case. The control signals are given by

$$\begin{aligned} u_1(t) &= \theta_{c1}(t)\cos(\omega t) - \theta_{s1}(t)\sin(\omega t)\ , \\ u_2(t) &= \theta_{c2}(t)\cos(\omega t) - \theta_{s2}(t)\sin(\omega t)\ . \end{aligned}$$

Define the vectors,

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix},\ y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix},$$

$$\theta_c(t) = \begin{bmatrix} \theta_{c1}(t) \\ \theta_{c2}(t) \end{bmatrix},\ \theta_s(t) = \begin{bmatrix} \theta_{s1}(t) \\ \theta_{s2}(t) \end{bmatrix},$$

so that the controller signal vector can be written as

$$u(t) = \theta_c(t)\cos(\omega t) - \theta_s(t)\sin(\omega t)\ .$$

The adaptive parameters $\theta_c(t), \theta_s(t)$ are updated according to

$$\begin{bmatrix} \dot{\theta}_c \\ \dot{\theta}_s \end{bmatrix} = -g\ G^{-1} \begin{bmatrix} y\cos(\omega t) \\ -y\sin(\omega t) \end{bmatrix},$$

where $g > 0$ is an arbitrary adaptation gain. An averaging analysis may be used to show that, for small enough g, the stability of the adaptive system is guaranteed if the 4×4 matrix G is chosen to be

$$G = \frac{1}{2}\begin{bmatrix} P_R & -P_I \\ P_I & P_R \end{bmatrix}$$

with

$$P(j\omega) = P_R + jP_I = \begin{bmatrix} P_{11}(j\omega) & P_{12}(j\omega) \\ P_{21}(j\omega) & P_{22}(j\omega) \end{bmatrix}.$$

The implementation of the algorithm requires the inverse of the 4×4 matrix, G, which is equal to

$$G^{-1} = 2 \begin{bmatrix} D^{-1} & P_R^{-1} P_I D^{-1} \\ -P_R^{-1} P_I D^{-1} & D^{-1} \end{bmatrix} ,$$

with $D = P_R + P_I P_R^{-1} P_I$, assuming that the matrix is invertible. In real-time implementation, the approximate inverse,

$$G^{-1} \simeq 2 \begin{bmatrix} (P_R^2 + P_I^2)^{-1} P_R & (P_R^2 + P_I^2)^{-1} P_I \\ -(P_R^2 + P_I^2)^{-1} P_I & (P_R^2 + P_I^2)^{-1} PR \end{bmatrix} ,$$

was used, which reduced the computational complexity. The inverse is exact if $P_R P_I = P_I P_R$, *i.e*, if the real and imaginary parts of the plant frequency response matrix are commutable. This property is satisfied in particular if

$$\begin{aligned} P_{11}(j\omega) &= P_{22}(j\omega) , \\ P_{12}(j\omega) &= P_{21}(j\omega) . \end{aligned}$$

These conditions may be viewed as "symmetry" conditions, because the behavior of the system is the same if the inputs and outputs are permuted. In the experiments, the geometric disposition of the speakers and microphones was symmetric, and the conditions were verified to be (approximately) true.

3. Experimental Results

The scheme was implemented on an experimental active noise control system developed at the University of Utah. The algorithm was coded in assembly language of Motorola DSP96002 32-bit floating-point digital signal processor. The sampling rate was set at 8 kHz. A single bookshelf speaker with a 4-inch low-frequency driver, located approximately 4 ft away from the error microphones, generated the periodic signal constituting the noise source. The signals were collected by two microphones separated by 2.7 ft. These signals were passed through an anti-aliasing filter and sampled by a self-calibrating 16-bit analog-to-digital converter before being sent to the DSP system. The controller output signals were sent to two noise cancelling speakers separated by 2.7 ft and located 3 ft in front of the microphones (or 1 ft away from the noise source).

The algorithm requires knowledge of the frequency response matrix $P(j\omega)$ of the plant for the frequency estimation as well as for the disturbance rejection. During an initial identifying stage, an estimate $\hat{P}(j\omega)$ was obtained, which was fixed and used in the control scheme afterwards. The frequency response at a given frequency ω_0 was determined by the empirical transfer function estimate (ETFE, [7]). Let the first input be a pure sinusoid of $\cos(\omega_0 n)$ and the second input be zero, $\hat{P}_1(j\omega_0)$, which is the first colomn of $\hat{P}(j\omega_0)$, was obtained through

$$\begin{aligned} \hat{P}_1(j\omega_0) &= \hat{P}_R + j\hat{P}_I , \\ \text{with } \hat{P}_R &= \frac{2}{N} \sum_{n=1}^{N} Y(n) \cos(\omega_0 n) , \\ \hat{P}_I &= -\frac{2}{N} \sum_{n=1}^{N} Y(n) \sin(\omega_0 n) , \end{aligned}$$

where $Y(n)$ is the vector of plant outputs, and $N = \frac{k\pi}{\omega_0}$ with $k = 1, 2, 3 \ldots\ldots$. The real and imaginary parts of the frequency response were obtained at 32 different frequencies, spaced between 90 Hz and 375 Hz, and the results were saved in a look-up table. In real-time, the frequency response at the estimated frequency was obtained by linearly interpolating the look-up table and the matrix G was adjusted accordingly.

Figure 4 and Fig. 5 show the signals from the error microphones #1 and #2, respectively. For the purpose of comparison, the control algorithm was not engaged until 3.0 sec. into the experiment. The frequency of the noise was 160 Hz. The figures show that the algorithm, once engaged, reduced the effect of the disturbances by approximately 20 dB (a factor of 10) within one second.

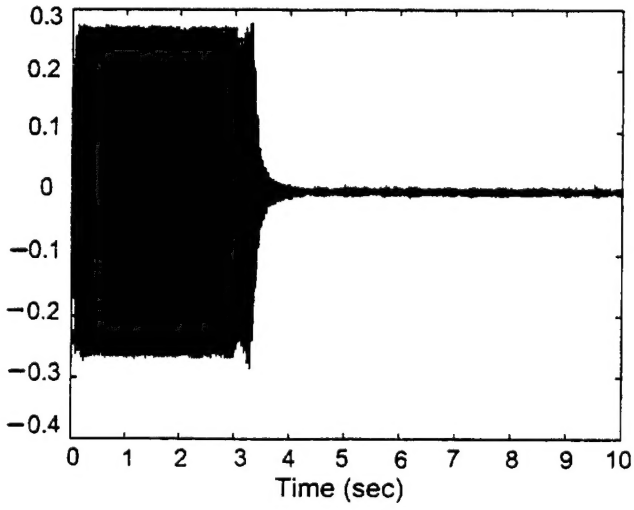

Figure 4: Error Signal From Microphone #1

The frequency estimate and two of the amplitude estimates are shown in Fig. 6 and Fig. 7, respectively. The initial values of the magnitude estimates were zero, and the initial frequency estimate was 100 Hz. The figures demonstrate the wide convergence region of the adaptive algorithm.

Figure 8-11 show the estimates of the plant frequency responses that were used by the adaptive algorithm. Fig.8 and Fig.9 are the magnitude response and the phase response of $P_{11}(j\omega)$, respectively, while Fig.10 and Fig.11

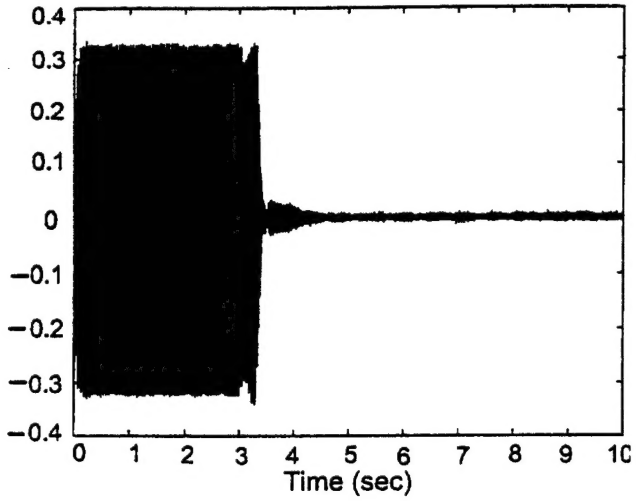

Figure 5: Error Signal From Microphone #2

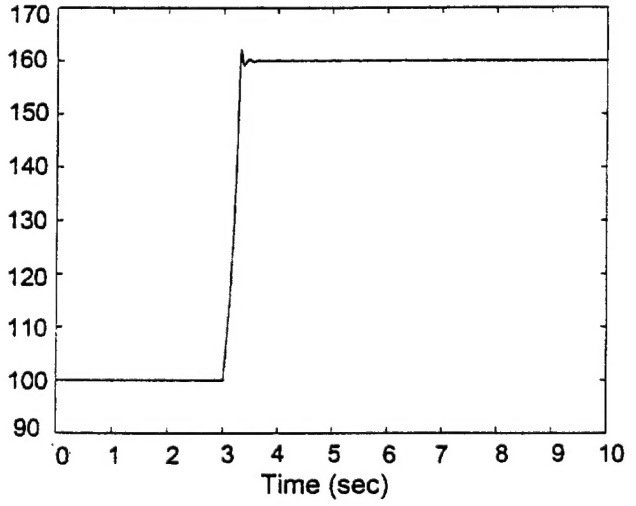

Figure 6: Frequency Estimate(Hz)

are the respective responses of $P_{12}(j\omega)$. The other elements of the plant transfer function matrix were assumed identical by symmetry and were not identified. This assumption need not be made in general, but if the assumption is satisfied, it may be used to reduce the complexity of the code. The magnitude responses show a significant number of peaks and valleys, which are not due to noise or inaccuracies in the system. Rather, they are repeatable and may be attributed to acoustic resonances in the small room that was used for experimentation. The phase responses mostly consist of the linear phase associated with the delay due to sound propagation from the speakers to the microphones. These frequency response plots are indicative of the significant control challenge posed by active noise cancellation in 3D space.

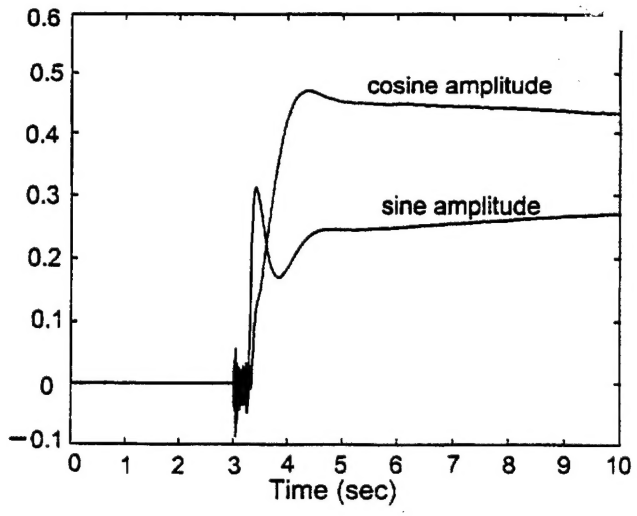

Figure 7: Amplitude Parameter Estimates

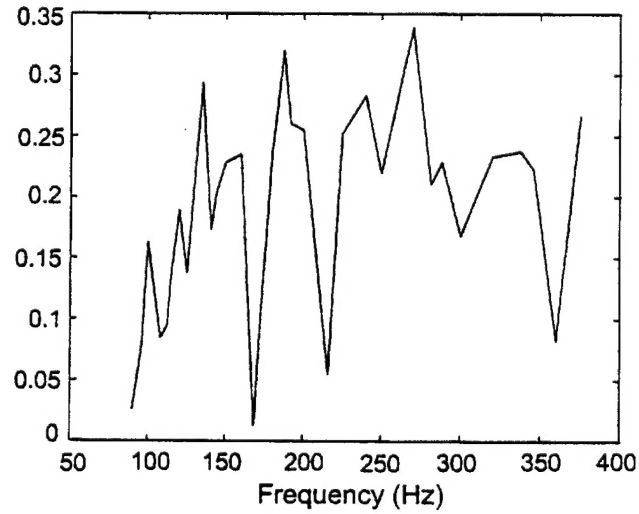

Figure 8: Magnitude Response of $P_{11}(s)$

4. Conclusions

An active noise control algorithm was proposed for multi-channel systems. The case of sinusoidal disturbances of unknown frequency was considered. General periodic disturbances may be rejected with this algorithm, but such rejection requires more complex computer codes, which has not been developed or tested at this time. The approach consists of estimating the

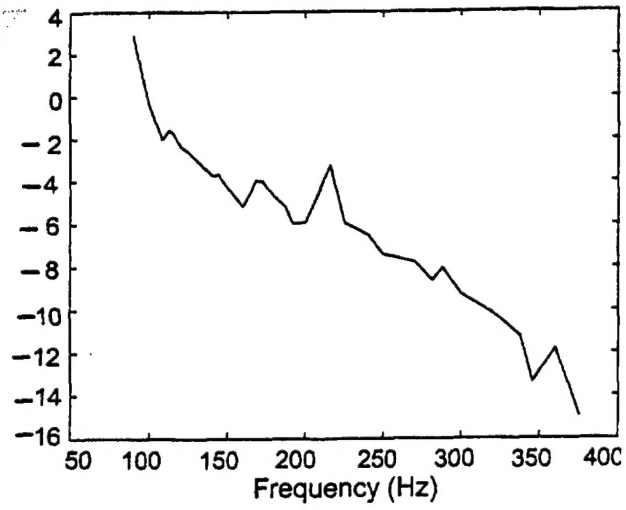

Figure 9: Unwrapped Phase Response of $P_{11}(s)$ (rad.)

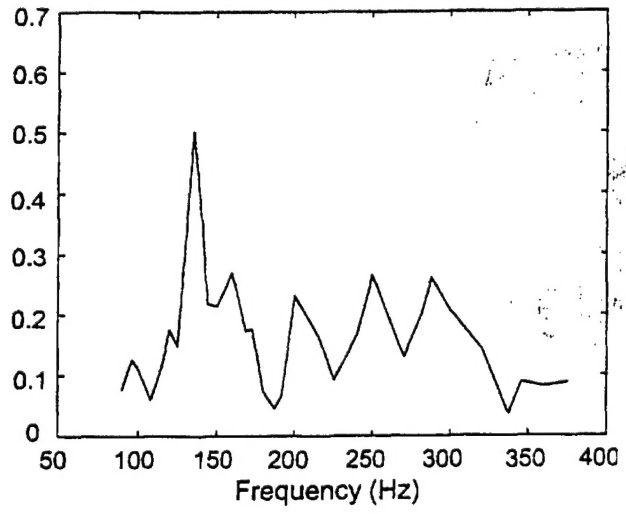

Figure 10: Magnitude Response of $P_{12}(s)$

frequency of the disturbance and using the estimate in a disturbance rejection scheme designed for the known frequency case. An important element is the use of a modified plant output for frequency estimation, so that the resultant signal does not vanish when noise reduction becomes effective. Experimental results showed the validity of the scheme proposed and the success of the algorithm in a dual channel active noise control system. The methods are not limited to active noise control problems, but may be applied to a wide range of control problems, where disturbances of unknown or varying frequency must be rejected.

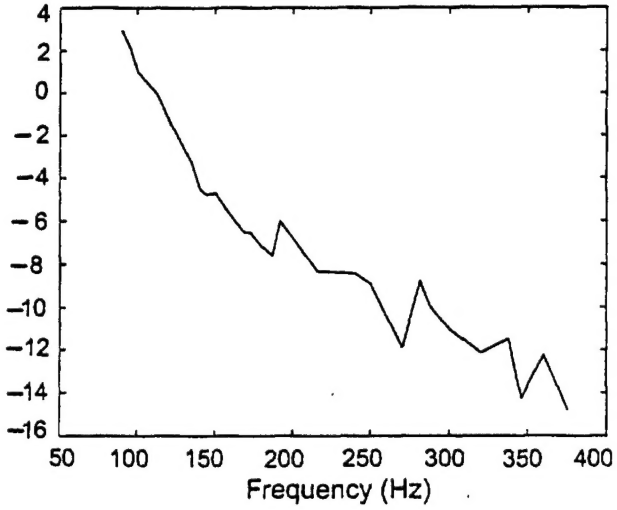

Figure 11: Unwrapped Phase Response of $P_{12}(s)$ (rad.)

5. References

[1] S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, New York, Wiley, 1996.

[2] M. Bodson & S. Douglas, "Adaptive Algorithms for the Rejection of Sinusoidal Disturbances with Unknown Frequency," *Automatica*, vol. 33, no. 12, pp. 2213-2221, 1997.

[3] M. Bodson, J. Jensen, & S. Douglas, "Active Noise Control for Periodic Disturbances," *Proc. of the American Control Conference*, Philadelphia, PA, pp. 2616-2620, 1998.

[4] M. Bodson, "Algorithms for Feedback Active Noise Control and Narrowband Disturbances," *Proc. of the ISMA23 (International Conference on Noise and Vibration Engineering)*, Leuven, Belgium, vol. 3, pp. 1279-1286, 1998.

[5] P.A. Regalia, "An Improved Lattice-Based Adaptive IIR Notch Filter," *IEEE Trans. Signal Processing*, vol. 39, no. 9, pp. 2124-2128, 1991.

[6] M. Bodson, A. Sacks and P. Khosla, "Harmonic Generation in Adaptive Feedforward Cancellation Schemes," *IEEE Trans. on Automatic Control*, vol. 39, no. 9, pp. 1939-1944, 1994.

[7] L. Ljung and T. Glad, *Modeling of Dynamic Systems*, Prentice Hall, Englewood Cliffs, New Jersey,1993